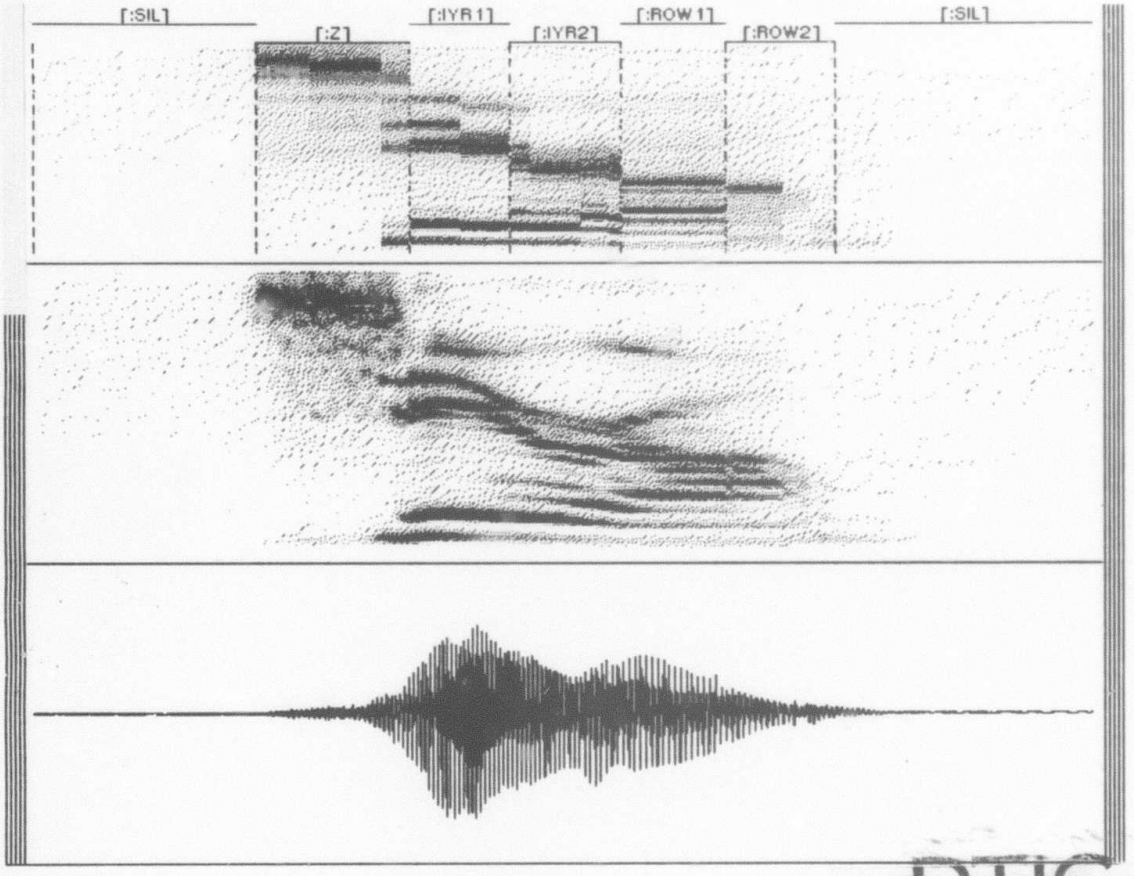
PROCEEDINGS

Speech Recognition Workshop

March 1987



Sponsored by:

DARPA

Defense Advanced Research Projects Agency
Information Sciences and Technology Office

SPEECH RECOGNITION

Proceedings of a Workshop
Held at
San Diego, California
March 24-26, 1987

Sponsored by the
Defense Advanced Research Agency

Science Applications International Corporation
Report Number SAIC-87/1644
Lee S. Baumann
Workshop Organizer

This report was supported by
The Defense Advanced Research
Projects Agency under DARPA
Order No. 5605, Contract No. N00014-86-C-0700
Monitored by the
Office of Naval Research

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SAIC-87/1644	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SPEECH RECOGNITION Proceedings of a Workshop, March 1987		5. TYPE OF REPORT & PERIOD COVERED Annual Technical February 1986 - March 1987 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Lee S. Baumann (Ed.)		8. CONTRACT OR GRANT NUMBER(s) N00014-86-C-0700
9. PERFORMING ORGANIZATION NAME AND ADDRESS Science Applications International Corporation 1710 Goodridge Drive, T-10-4 McLean, Virginia 22102		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order No. 5605
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE March 1987 13. NUMBER OF PAGES 123 pps.
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech Recognition; Signal Processing; Acoustic Phonetics; Intelligent Systems; Cochlear Models; Speaker-independent vowel recognition; Pitch tracking; Vowel Coarticulation; Phonological studies; Continuous word recognition; Robust HMM-based techniques for speech recognition.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document contains the technical papers for the Speech Recognition Program which were reviewed by the key research specialists from the research activities participating in the program sponsored by the Information Science and Technology Office, Defense Advanced Research Projects Agency. The reviews of these papers were presented at a workshop conducted on 19-20 February 1986, in Palo Alto, California.

Keywords: reports; abstracts;

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

Page

TECHNICAL PAPERS (Cont'd)

TABLE OF CONTENTS (Cont'd)

Page

TECHNICAL PAPERS (Cont'd)

FOREWORD

The semi-annual workshop for research and government personnel involved in the DARPA program on Speech Recognition was held in San Diego, California on 24-26 March 1987. The purpose of the workshop was to review progress on research efforts undertaken over the past year by the participating organizations; Carnegie-Mellon University, BBN, MIT, SRI International, National Bureau of Standards, MIT Lincoln Laboratory, Texas Instruments, Dragon Systems, and Schlumberger. Also participating were representatives from DARPA, SPAWAR, NSA, NOSC, RADC, AFWAL, Xerox Research Laboratory and Signition, Inc.

In his opening remarks, Commander Sears, the DARPA Speech Recognition Program manager, advised the group that in addition to the site progress reports, two important items would receive considerable attention; details related to performance evaluation and database needs for the October demonstration. Other items covered included results from the March dress rehersal, update plans for Phase II, and a variety of issues including the role of speech understanding for future systems and plans for the speech program to interact with the strategic computing architecture and natural language programs.

This proceeding consists of technical reports which were reviewed by the key individuals for that program at the workshop. The papers are arranged generally in accordance with the order of presentation.

The last day of the workshop consisted of a visit to the research facilities of the Naval Ocean Systems Center. This site visit was arranged and hosted by Ms. Elaine Schiller and Mr. Steve Nunn, members of the NOSC research staff. Several programs being developed for use by Naval Forces which utilize speech processing, natural language, or knowledge based systems were demonstrated and explained to the group by NOSC personnel. The demos proved to be helpful to the group to understand and relate technology developments to real world problems.

The figure used for the cover design was provided by Richard Lyon of the Slumberger Palo Alto Research Laboratory. Dr. Lyon states that the figure shows three representations of the spoken digit "zero." The bottom graph shows the time domain wave form; the middle section is a cochleogram which is a representation of the processing in the ear; the top section shows the segmentation and

recognition of the digit using the scale-space technique.* Thanks are due to Mr. Tom Dickerson of the SAIC graphics department for the layout of the cover of the proceedings and to Ms. Dianne Williams for assistance in putting the report together.

Lee S. Baumann
Science Applications
International Corporation
Workshop Organizer

* More information is available in Lyon and Loeb's paper contained herein.

AUTHOR INDEX

TECHNICAL PAPERS

The lexical access component of the CMU continuous speech recognition system.

Alexander I. Rudnicky, Zong-ge Li, and Lynn K. Baumeister

Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213.

Abstract

The CMU Lexical Access system hypothesizes words from a phonetic lattice, supplemented by a coarse labelling of the speech signal. Word hypotheses are anchored on syllabic nuclei and are generated independently for different parts of the utterance. Junctures between words are resolved separately, on demand from the Parser module. The lexical representation is generated by rule from baseforms, in a completely automatic process. A description of the various components of the system is provided, as well as performance data.

This paper describes the lexical access system under development at Carnegie-Mellon University. The design of the hypothesizer is based on the following principles:

- ***Words can be generated bottom-up with a very high degree of accuracy.*** Given a sufficiently accurate transcription of the speech signal, it is possible to use a completely bottom-up paradigm to drive word recognition, without assistance from higher-level constraints, such as those that might be provided by a narrowly defined task, or restrictive grammar.
- ***Multiple knowledge sources are necessary for generating high-quality word hypotheses.*** The information contained in a phonetic transcription is of itself insufficient to guarantee high accuracy, additional constraints on interpretation, either derived from alternate analyses of the signal, or from stored knowledge about speech characteristics are necessary for accurate hypothesizing.

The word hypothesizer produces lexical hypotheses using the phonetic label lattices produced by the Acoustic-Phonetic component of the system [1]. Figure 1 presents a schematic diag m of the hypothesizer module. The principal functional components of the word hypothesizer are the following:

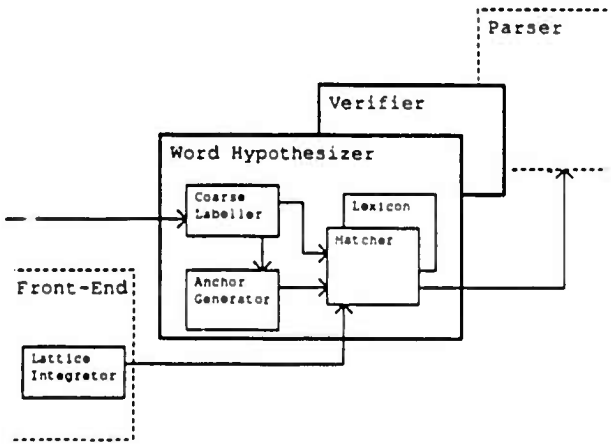
- **Matching Engine:** The matcher generates a lattice of word hypotheses. A modified beam-search algorithm is used to match a phonetic transcription against a lexicon stored in the form of a phonetic network.
- **Anchor Generator:** The matcher does not attempt to match words at all possible positions in an utterance, as might, for example, a two-level DP algorithm. Rather, the anchor generator uses a coarse segmentation of the speech wave to locate syllable nuclei and to define likely word regions ("anchors").
- **Coarse Labeller:** It is capable of producing a robust segmentation of the speech signal into silence, noise and vocalic regions. Coarse labels are used both to locate syllable nuclei and as a secondary source of information by the matcher.

In addition to the above components, the lexical access system also includes a **Phonetic Lattice Integrator** and a **Juncture Verifier**.

The **Phonetic Lattice Integrator** combines and transforms the independently generated information contained in the stop, closure, vowel, and fricative lattices produced by the Acoustic-Phonetic labelling component. The actions performed by the Phonetic Lattice Integrator include the adjustment of boundaries, the resegmentation of overlapping segments, and the combination of label probabilities from different lattices.

The role of the **Verifier** is to process word-juncture verification requests generated at the Parser level. The Verifier analyses junctures between words and indicates to the Parser whether the words in question can form a phonetically acceptable sequence.

Figure 1: Word Hypothesizer system diagram



1 Matching Engine

Words are hypothesized by matching an input sequence of labels against a stored representation of possible pronunciations, the *lexicon*. The matching algorithm makes use of both a phonetic lattice and a coarse lattice. The network search algorithm used in the current system is based on the beam

search algorithm, but has been substantially modified to deal with the particular demands of the current task.

Beam-search is a modified best-first search strategy that extends paths with scores within some window of the global best score. The width of this window (the "beam") controls the severity of pruning applied to the search. The principal difference between a conventional beam-search (as implemented, e.g., in the HARPY system [3]) and the current algorithm is the ability to simultaneously search paths of different lengths. Although search tree is expanded segment by segment (i.e., is time-locked), paths may begin at a number of separate locations in the anchor region (see below). Because of the resulting differences in path lengths, the bounds of the beam cannot be calculated in a simple fashion. The solution used is to normalize all path scores by their duration.

The size of the search tree is controlled in two ways, by modifying the width of the beam and by altering the score of a given path through the use of penalties.

Beam width is calculated dynamically at each ply and is based on a pre-set width modified by a value based on the size of the expansion stack generated at the preceding ply. The effect is to relax pruning when there are few nodes on the stack and to tighten it when the stack begins to grow excessively large. One practical consequence of this is to allow paths that initially have poor scores to survive long enough to accrete positive evidence. Another consequence is to permit more severe pruning later in the search when the number of path is typically the greatest. Dynamic beam adjustment speeds the algorithm up by 39%, and reduces the depth of the output lattice by 18%, while maintaining match accuracy.

In addition to pruning based on beam width, the system uses several other strategies to control the size of the search tree. Since search progresses uniformly through successive segments, paths that pass through the same node in the network at the same segment ("collisions") are compared, and only the best path is kept (work with HARPY has shown that although this is a sub-optimal strategy, it nevertheless, in practice, produces near-optimal network traversals, at substantial savings in search effort).

Two additional pruning factors come into play through their ability to modify the cost of a path and thereby place it outside the search beam.

The first of these is a duration range associated with each phonetic label in the lexicon. Paths that remain in a particular state (phone) for either shorter or longer than the characteristic range for that phone incur a penalty. For example, the duration range for a /b/ is [3 30], based on the observation that /b/ bursts typically do not exceed 20ms. the constraint for an /s/ is [50 250], again based on the observation that /s/ phones are typically at least 40ms in duration. Similarly, the duration constraint provides a different range for an /ax/ as opposed to a diphthong, such as /a^{y}/. Exiting a state either too early or too late incurs a penalty, this penalty is added to the path score.

A second type of penalty is assessed when the coarse class of a phone mismatches that provided by the coarse labeller. The assumption here is that if the two types of label do not match, an error is likely. Again, the penalty added to the path score makes it a candidate for pruning. If the match is already poor, this penalty hastens its pruning. In fact, this penalty is most useful for rapidly terminating paths that wander across category boundaries, for example, remaining in a vocalic state when the segments have become non-vocalic. In the current implementation, enforcing cross-lattice consistency reduces the size of the search by a factor of about 3. If consistency were absolutely enforced (i.e., inconsistency results in immediate pruning) search would be reduced by a factor of 6-7 though with a loss in accuracy.

The calculation of the path score is performed according to the following formula:

$$\frac{\sum_{i=1}^{n} d_i(k \log p_i)}{\sum_{i=1}^{n} d_i} + \frac{\sum_{i=1}^{n} a_i P_D}{\sum_{i=1}^{n} d_i} + \frac{\sum_{i=1}^{n} d_i P_L}{\sum_{i=1}^{n} d_i} + q(S_E - S)$$

The formula consists of three terms: the phonetic score, the duration penalties, and the lattice mismatch penalties; n is the length of the path. The phonetic score consists of the following: d_i is a segment duration, p_i is a label probability, and k is a scaling factor (a computational convenience)

The duration penalty consists of a_i, the amount of discrepancy, and P_D, a system parameter controlling the degree of penalty. The lattice penalty consists of a system parameter, P_L, scaled by the duration of the segment, d_i. Normalization is necessary, as paths of different length need to be comparable. The final term in the equation represents a state shortfall. Each hypothesis in the lexicon is required to match a minimum number of core phonetic states. Matching less than this number implies that word has been severely reduced, a condition which is penalized in the current system.

2 The Lexicon

The lexicon is stored in the form of a phonetic network. The process of creating a net is as follows: For the chosen vocabulary, a set of base-form pronunciations is obtained. The sources of pronunciations that have been made use of include the following: lookup in an on-line phonetic dictionary, such as the Shoup dictionary, the generation of pronunciations using a letter-to-sound compiler (the MITalk system), or direct construction. Each approach has its advantages and disadvantages. We have found that automatic generation as a first pass, followed by hand correction, generally produces the most acceptable result and does so in a reasonable amount of time. Baseforms are further expanded into pronunciation networks in order to take into account different possible realizations of a word, such as those due to rapid-speech phenomena and coarticulatory effects. Possible variations in pronunciation are expressed in the form of phonological rules that are applied automatically (in an off-line procedure) to the baseform pronunciation. Figure 2 shows a typical rule, governing /ty/ desyllabification. The rule-applier scans the pronunciation string for the pattern specified in the FIF portion of the rule, binding the elements of the pattern as specified. Terms headed by a "+" match 0 or more elements, which are bound to the following variable (e.g., `LeftContext`). Terms headed by ">" must match a single element, typically meeting the constraints specified in the remainder of the clause; constraints are expressed in terms of phonetic features, such as CONS (consonant) or VELAR (a place of articulation). The THEN part of the rule has two clauses, the first specifies the portion of the pronunciation string to be emitted, the second clause the portion to be rescanned with the pattern. Depending on what is put into each clause, a rule may be made to apply once, multiple times, or iteratively to a pronunciation. The current CMU lexicon is constructed using a base of over 150 rules, covering several types of phenomena, including coarticulatory phenomena and front-end characteristics. A small number of additional rules perform necessary bookeeping functions.

Figure 2: A phonological rule

```
(IY-syl-loss-a
  (FIF ( (+ LeftContext)
         (> Tm1 (has CONS) )
         (> Tar (has VOWEL HIGH FRONT) (lacks LAX))
         (> Tp1 (has VOWEL))
         (+ RightContext)
         )
   THEN
       ( (LeftContext
          Tm1 (alt 'Y 'IY) Tp1)
         (RightContext))
       )
)
```

The above rule applied to the word COLUMBIA:

```
        K AX L UH M B IY AX
---->
        K AX L UH M B (Y , IY) AX
```

Expansion is performed by adding nodes and arcs to the base pronunciation through the application of phonological rules. The individual nets produced in this fashion are then merged together into a single network, the representation used by the matcher. The merge collapses common initial states to eliminate redundant matches and produces a network that fans out from few initial states into a larger number of states, the penultimate states corresponding to individual lexical entries.

3 Anchor Generation

The structure of speech constrains the possible locations of words in an utterance, that is, a word may not begin or end at some arbitrary point; permissible end-points are governed by the acoustic properties of the signal. To eliminate unnecessary matches, the system uses syllable anchors to select locations in an utterance where words are to be hypothesized. The anchor selection algorithm is straightforward and is based on the following reasoning. Words are composed of syllables, syllable all contain a vocalic center (de-voiced syllables can be treated as a special-case). Word divisions cannot occur inside a vocalic center, thus all syllable and word breaks will occur in the regions between vocalic centers. The coarse labeller provides information about vocalic, non-vocalic, and silence regions, as well as information about energy dips within vocalic regions (typically corresponding to liquids, glides, and nasals). This allows the utterance to be segmented into two regions: vocalic centers and boundary regions. An anchor, as used by the matcher, consists of two anchor regions, a beginning and an ending one, separated by one or more vocalic centers, the number of centers determining the number of syllables that words hypothesized for that region should have. Figure 3 provides a schematic diagram of the anchoring process.

The matching algorithm allows words to begin anywhere in the beginning region (i.e., the initial state of the network is put on the stack for each phonetic segment in this region). Paths may not transition into the the network's final state until path extends into the ending region. The algorithm is implemented in such a fashion that, for a given word in the lexicon, only a single, "best" hypothesis will be generated, where best means the lowest cost traversal through the lexical network.

Anchors have been used in the system in two modes *single-anchor* and *multiple-anchor*. In the single-anchor mode, anchors of different lengths are generated and the matcher is invoked separately for each one, as shown in (b). It should be apparent that this procedure, although simple, is inefficient. There two reasons for this: The entire network is applied to each anchor, thus time is wasted trying to force, e.g., 5-syllable words into 1-syllable anchors. Second, the same region of speech is scanned repeatedly, with the results of one scan being unavailable to subsequent scans. The multiple-anchor strategy alleviates these problems, at only a slight increase in algorithm complexity, by using anchors with multiple end regions. In this case, paths for words of inherently different durations can terminate at compatible points in the anchor and are not forced into inappropriate regions. A multiple-anchor strategy reduces computation by a factor of 3, while reducing the number of hypotheses generated by 60% (inappropriate mappings of words into syllables are eliminated). A third strategy is possible, though at this time has not been implemented. This is the use of *continuous anchors*, where each inter-vocalic region serves both as an entry point and and end-region for the search (d). The advantage of a continuous anchor strategy is that it allows the simultaneous comparison of paths that span different portions of the signal. The quality of input, however, determines the success of this strategy.

4 Coarse Labeller

The coarse labelling algorithm is based on the ZAPDASH algorithm [2], modified to generate additional labels and to provide a more accurate segmentation of the signal. The coarse labeller codes the speech signal using four parameters extracted on a centisecond basis, these being peak-to-peak amplitude and zero-crossing counts for low-passed and high-passed portions of the signal (the crossover being at 1 kHz). Segments are located by seeking frames characteristic of a particular energy type using a strict criterion (an "anchor"), then expanding these into a region using a laxer criterion. In addition to the anchor-extend procedure, rules are used to apply contextual information to ambiguous regions and to perform boundary adjustment.

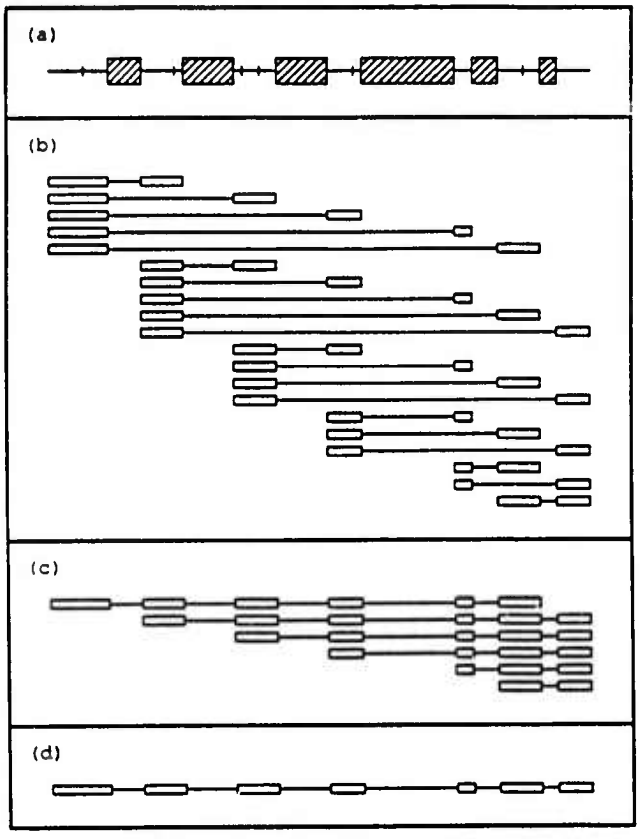
The algorithm currently distinguishes the following acoustic events: silence, including "true" silence and noisy silence; sonorants, including vocalic centers as well as inter-vocalic sonorant energy dips (such as nasals or liquids); a variety of aperiodic signals, corresponding to fricatives, aspirates, etc.

The algorithm is robust and speaker-independent, and operates reliably over a large dynamic range. Currently, the quality of coarse-labelling is such that less than 0.1% of syllabic nuclei are missed. A number of extra nuclei are generated, though this does not create difficulties for either anchor generation or lattice cross-checking during matching.

5 Phonetic Lattice Integrator

The phonetic labels produced by the front-end [1] are grouped into four separate lattices: vowels, fricatives, closures, and stops. Moreover, labels both within and between lattices may overlap in time. The role of the integrator is to combine these separate streams and produce a single lattice consisting of non-overlapping segments, each segment containing the information from one or more segments in the original lattices. The integrator maps the label space used by the front-end into the label space used in the lexicon. For example, the label "stop" is expanded into the appropriate set of lexical labels ([ptkbdg]). In addition, the integrator uses a confusion matrix to partition the probability assigned to a front-end label into several labels that it may be confused with, thus an input iy label will be reflected in not only the lexical IY label, but also the IH label.

Figure 3: Anchor region selection



Notes: (a) A coarse segmentation of the speech signal. The hatched blocks are vocalic centers. (b) Single anchors for the signal in (a), a total of 20 anchors. Search can begin from any segment in the onset region, must proceed through the middle, and can terminate in the coda region. (c) multiple anchors, search can begin in the first region and end at any subsequent region. (d) continuous anchors, search can begin in any but the last region and end end at any but the first region.

The use of a confusion matrix to map the input symbol produces an improvement in accuracy, but at the cost of additional search. For the 708 word Shipping Management task, first choice accuracy goes from 32% to 42%, while the average number of states examined per word rises 2.5-fold, from 958 to 2381. We believe that the advantage of this transformation is due to the ability of the confusion matrix to capture the broad behaviour of classifier labels across different contexts and thereby supplement the probabilities generated for a given classification region (see [1])

6 Verifier

Words are hypothesized in isolation, that is, without regard to any sequential constraints between words. In this sense, the system is completely bottom-up, since no syntactic, semantic, or task constraints are brought to bear on the process of hypothesization. The resulting word lattice consequently contains many potential sequences of words. The parser [4] attempts to construct plausible sequences, but does not have the information necessary to decide whether a particular sequence is phonetically acceptable. The Verifier examines junctures between words and determines whether these words can be connected together in a sequence. The verifier deals with three classes of junctures: *abutments*, where two words join together without overlap or intervening segments; *gaps*, where the two words are separated, and *overlaps* where the words share one or more segments. In general, overlaps that involve inconsistent interpretations of the speech signal are disallowed, and gaps that contain significant speech events are also disallowed. Figure 4 shows the distribution of juncture types for the Email task (considering only correct word sequences), together with Verifier accuracy.

Figure 4: Juncture types and Verifier performance

Juncture type	Incidence	(rejection)
Abuts	51%	(0%)
Gaps	20%	(1.7%)
Overlaps	29%	(5.9%)

7 Performance

System performance was evaluated by calculating the rank of the correct word for a known anchor position. This metric is somewhat conservative, since words with the same core but with different endpoints are compared (for example, the embedded word END competes with the word SEND under the current scheme). Figure 5 gives performance for two types of input data, spectrogram reading and automatic labelling (using the September 1986 CMU system). The task is the 324 word Electronic Mail task.

Figure 5: Word Matcher performance

	Spectrogram	Automatic
1st choice	60%	32%
Top 3	83%	55%
Top 10	93%	76%

8 Discussion

The CMU lexical access system operates as a word-spotter, generating all likely hypotheses, anchored on syllable nuclei. The design of the matching algorithm demonstrates the appropriateness of a unified matching strategy, as opposed to a strategy that uses coarse-filtering of word candidates followed by fine-grain phonetic matching: Coarse-class constraints are used as a component of the pruning strategy and do not entail the use of hard decisions implicit in, e.g., a filter design. This approach provides a maximum of flexibility to subsequent levels of processing.

Experience with the anchor-based matcher has revealed a number of shortcomings in its design. For example, the benefits of anchoring are only realized when syllables are correctly detected. Failure to identify a syllable boundary can be catastrophic—one or more words may be lost as a result. Similarly, the word-spotting mode in which the system operates makes it difficult to make use of constraints that could be imposed across word boundaries and, moreover, complicates the process of interpreting juncture phenomena. Given these findings, we have begun to explore a different approach to word matching. The new algorithm is not based on anchoring and it incorporates explicit modeling of juncture phenomena. We refer to the new algorithm as a ***rolling*** matcher, as it "rolls" through an utterance rather than jumping from anchor to anchor.

In order to avoid the compromises made in the lattice integration step, the system has changed to use input in the form of a ***phone network***. A phone network performs the same function as the lattice integrator—it coordinates the output of the four different acoustic-phonetic modules. Specifically, it provides segment boundary alignment by coercing segment endpoints, it resolves conflicting overlap conditions (i.e., by providing alternate paths), and it ensures that all regions of an utterance can be traversed (i.e., by labeling regions not labeled by any of the primary modules according to their coarse-class labels). Another benefit of a phone network representation, from an acoustic-phonetic point of view, is that it allows correct handling of sequential dependencies (e.g., the influence of liquids on vowel color).

In contrast to the compilation process described earlier in this paper, network compilation is now performed in two separate passes. The first generates intra-word variations, producing *sub-nets* for each baseform in the lexicon. After these sub-nets are merged into a single net, a second set of rules is applied to generate correct cross-word connections, dealing with such varied phenomena as gemination, insertion (e.g., of glides), and deletions (e.g., of closures).

The matching process consists of "rolling" the (lexical) network through the phonetic network. Successful paths through the lexical network (i.e., traversal from a given start node to a given end node produces a word hypothesis. The word hypothesis is placed on the output lattice, and matching continues on to all words that can legally follow the word that was just completed.

Early analyses indicate that the Rolling matcher differs from the Multiple-Anchor matcher in several respects: The word lattice produced by the Rolling matcher is substantially denser than the one produced by the multiple-anchor matcher. This is because the latter produces a single best match for a given region of speech, the former produces multiple matches, with different end-points. This property is actually desirable, as it simplifies the juncture-validation problem—multiple end-points allow the parser to select the optimal version of a hypothesis, without the need for detailed juncture analysis.

9 Acknowledgement

The research described in this paper is sponsored by the Defence Advanced Projects Agency under contract N00039-85-C-0163. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defence Advanced Research Projects Agency or of the United States Government.

10 References

[1] Cole, R., Phillips, M., Brennan, B., & Chigier, B. The CMU phonetic classification system. In *Proceeedings of the ICASSP*. 1986.

[2] Gill, G., Goldberg, H., Reddy, R., & Yegnanarayana, B. *A recursive segmentation procedure for continuous speech*. Technical Report, Carnegie-Mellon University, May, 1978.

[3] Lowerre, B. and Reddy, R. The HARPY speech understanding system. *Trends in speech recognition*. Prentice-Hall, 1980.

[4] Stern, R.M, Ward, W.H., Hauptmann, A.G., & Leon, J. Sentence parsing with weak grammatical constraints. In *Proceedings of the ICASSP*. 1987.

SENTENCE PARSING WITH WEAK GRAMMATICAL CONSTRAINTS

Richard M. Stern, Wayne H. Ward, Alexander G. Hauptmann, and Juan Leon

Department of Computer Science
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

ABSTRACT

This paper compares the recognition accuracy obtained in forming sentence hypotheses using several parsers based on different types of weak statistical models of syntax and semantics. The inputs to the parsers were word hypotheses generated from simulated acoustic-phonetic labels. Grammatical constraints are expressed by trigram models of sequences of lexical or semantic labels, or by a finite-state network of the semantic labels. When the input to the parser is of high quality, the more restrictive trigram models were found to perform as well as or better than the finite-state language model. The more restrictive trigram and network models of language produce better recognition accuracy when all correct words are actually hypothesized, but strong constraints can degrade performance when many correct words are missing from the parser input.

INTRODUCTION

It is well known that the accuracy of automatic speech recognition systems can be greatly improved by the imposition of syntacic, semantic, and grammatical constraints. These constraints have typically been expressed in the form of finite-state or phrase-structure network models [1, 2, 3] and second order Markov models (*e.g.* [4]). We would generally expect that more specific domain-dependent constraints could provide a greater improvement of recognition accuracy, but weaker grammatical constraints may prove advantageous if the input to the sentence hypothesizer is noisy or if extragrammatical utterances are frequently encountered.

Carnegie Mellon University is presently developing a large-vocabulary speaker-independent speech recognition system. The system includes a feature-based acoustic-phonetic hypothesizer [5], an island-driven word hypothesizer [6], and several sentence parsers that convert the outputs of the word hypothesizer into sentence candidates.

We have explored several schemes for representing syntactic and semantic knowledge in these parsers, including case frames [7] and simple statistical models of sequences of syntactic and semantic categories of the word candidates. Most of the statistical grammars make use of a second-order Markov model to represent local syntactic and semantic phenomena. Our work differs from most other language models employing this "trigram" representation (e. *g.* [4]) in that constraints are expressed in terms of probabilites of sequences of lexical or semantic labels or "tags", rather than the individual words in the vocabulary themselves. In addition to providing reasonable accuracy, we also believe that this approach is a promising way to reduce the amount of storage and training required to effectively model word usage in tasks with very large vocabularies. A small number of other groups have also proposed trigram models using a reduced number of syntactic or semantic tags but these groups have not dicussed the range of language models and input conditions that will be considered here.

The purpose of this paper is to compare the ways in which the degree of specificity of the grammatical constraints affect the recognition accuracy obtained with a deterministic finite-state network representation of the task and with some of the probabilistic trigram grammars, considering inputs to the sentence parsers of varying quality.

In the following sections we first describe the manipulations of the input to the sentence parsers. We then briefly describe the different parsers that are used in the present study. Finally, we compare the recognition accuracy of these parsers in the presence of the different types of degraded input and comment on some of the implications of our results.

WORD LATTICES USED IN EXPERIMENTS

For each sentence presented to the CMU recognition system, the word hypothesizer outputs a large number of candidate words, which are each characterized by a begin time, an end time, and an acoustic-phonetic plausibility score. This set of annotated word hypotheses is referred to as the "word lattice" of the input sentence.

The expected word accuracy of sentences produced by a parser is closely related to two major attributes of the word lattice: (1) the relative acoustic-phonetic scores of the correct words that are present on the lattice, and (2) the percentage of correct words that are missing from the lattice.

In order to compare the effects of degraded word quality and omissions from the word lattice, we prepared six sets of lattices in which the relative scores of correct words and percentages of missing words were artificially manipulated. The characteristics of these sets of lattices, which were used in our performance calculations, may be summarized as follows:

- **Original lattices** - 48 sentences from a 325-word electronic mail (email) task were recorded by three female and two male speakers. These sentences contained a total of 281 words. A set of acoustic-phonetic labels was created manually by expert spectrogram readers from spectrograms and other visual displays of the digitized waveforms. This labelling was "blind" in that the labellers did not know the identity of the correct utterance. Since these lattices nominally represent "ideal" output from the acoustic-phonetic module, they are useful for evaluating degradations in recognition performance introduced by the system's word and sentence hypothesizers. Word lattices were generated from the acoustic-phonetic labels in the fashion described in [6].
- **High-quality lattices** - These lattices are the subset of the 48 original blind-labelled word lattices that have no correct words missing and no incorrectly penalized word junctures (see below). There are 31 sentences with 172 total words in these lattices. The remaining sets of word lattices were obtained by artificially degrading these word lattices.
- **Degraded-quality lattices** - **Moderately-degraded** and **severely-degraded** word lattices were created by adding a constant to the acoustic-phonetic scores of words in the **high-quality lattices**. This had the effect of worsening the

scores of the correct words relative to the scores of the incorrect words.

- **Missing-word lattices** - **Missing-word lattices** were created by randomly deleting either 10 or 25 percent of the correct words from the **high-quality lattices**.

The overall quality of these lattice is summarized in Figure 1. Each curve of Figure 1 shows how many words in the lattice per correct word need be examined to ensure that a given percentage of correct words is included. For example, Figure 1 shows that the **high-quality lattices** and the **moderately** and **severely-degraded lattices** contain approximately 100 percent of the correct words (if we are willing to consider a sufficiently large number of incorrect words as well), while the lattices with **missing words** contain no more than 70 and 85 percent of the correct words, no how many words are examined. (The asymptotes in these three curves differ slightly from their nominal values because of differences in the word-boundary criteria used by the hand labellers and the automatic lattice evaluation algorithms.)

PARSERS USING TRIGRAMS AND NETWORKS

We compared the word accuracy of a number of different left-to-right parsers in processing the various types of word lattices described above. These parsers make use of the same architecture, differing only in the types of knowledge used to evaluate candidate phrases. We will therefore first describe the overall structure and then describe each individual parser in terms of its use of syntactic and semantic knowledge.

As noted above, each parser receives a lattice of words that contains the begin time, end time, and a score for each word. It forms phrases from the words in left-to-right fashion. New phrases are created by attempting to add new words to the end of existing phrases. A beam search is used to prune the set of phrases retained for further expansion so that at any point in the parsing process only the 100 best-scoring phrases are retained.

The following types of knowledge are considered when auding a new word to a candidate phrase:

- **Word score** - The word score represents the likelihood for the word based on acoustic-phonetic evidence, provided by the word hypothesizer.

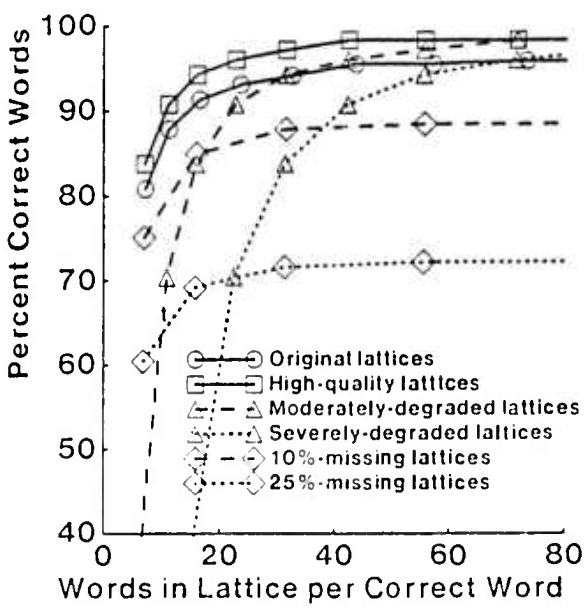


Figure 1: Comparison of quality of word lattices used in the sentence parsing experiments.

- **Word-juncture quality** - The quality of the acoustic-phonetic juncture between two words is scored by the junction verifier in the word hypothesizer, based on tables of penalties for overlaps and gaps.
- **Syntactic and semantic information** - Two different methods were used to score the syntactic or semantic plausibility: a finite-state network derived from a formal description of the grammar of the task and trigrams of frequencies of syntactic and/or semantic word classes.

The score for a phrase is a linear combination of the scores provided by each of the above knowledge sources. The weights used to combine these scores were determined parametrically from training data, and the recognition accuracy of the parsers is relatively insensitive to their exact value.

We now describe the various parsers in more detail.

Allword parser. This parser makes use only of acoustic scores and word juncture information in forming its phrase hypotheses, so any word can follow any other word.

Trigram parsers. The trigram measure is derived from the conditional probability of observing the syntactic or semantic classes of three words in sequence in a set of training sentences, which in turn is used to estimate the joint probability that the syntactic or semantic structure of the sequence of three words is correct. The overall utility of this approximation depends on the degree of domain specificity of the training sentences and syntactic classes used.

For each set of syntactic and/or semantic constraints, words are sorted into categories of one or more tags. Special tags are used to represent the beginning and ending of a sentence. When a word is added to the end of a phrase, it is assigned a trigram score based on the conditional probability of observing its tag given the two previous tags in the phrase.

We examined the following trigram parsers, which are identified by the types of syntactic and semantic knowledge that constrain their hypotheses.

- **Syntactic trigram parser** - In addition to word scores and word juncture information, this parser also incorporates syntactic information through trigrams of sequences of 41 tags denoting lexical categories. These tags were a subset of the approximately 90 lexical tags adopted by the compilers of the Brown corpus [8]. They include expanded designations of parts of speech, complete conjugations of some important verbs such as *be, do,* and *have, etc.*
- **Augmented trigram parser** - This parser is similar to the syntactic parser, except that a set of 55 tags is used. This set is somewhat more specific to the email task than the tags used by the compilers of the Brown corpus. For example, different designations are used for nouns representing people, places, and things, and there is a greater number of tags that designate classes of prepositions. Hence these tags describe a modest amount of semantic knowledge. We believe that these tags could eventually represent the syntax of a database-query system for an arbitrary task domain.
- **Semantic trigram parser** - The semantic trigram parser is similar to the syntactic parser, except that a set of 92 tags is used that corresponds to the labels of the nodes of the semantic network parser described below. These tags, and their trigram probabilities, are much more domain dependent.

Semantic Network Parser. The deterministic semantic network parser is derived from a description of the email task expressed in the form of case frames and simple phrase structure rules [9, 7]. These were manually combined into a semantic grammar of about 350 rules. The grammar was then compiled into a finite-state network similar to a Harpy network [1], for faster processing. This type of network can provide a semantic interpretation of the input utterance as well as mere word recognition. There were 92 different categories of words in the network, reflecting the semantic specificity of the encoding. The grammar encoding was *tight* in the sense that only grammatical sentences are accepted.

EXPERIMENTAL RESULTS AND DISCUSSION

Each of the parsers was run on each of the sets of lattices. Results are expressed by the percentage of correct words detected and by the percentage of incorrect words inserted by the parser. (The insertion percentage in this paper is defined to be the number of incorrect words found divided by the number of words uttered. Note that substitution errors cause both a decrease in the detection percentage and an increase in the insertion percentage.)

	Allword Parser	Semantic Trigram Parser	Semantic Network Parser
DETECTION PERCENTAGES:			
Original lattices	58	87	83
High-quality lattices	59	98	92
INSERTION PERCENTAGES:			
Original lattices	49	9	18
High-quality lattices	45	3	7

Table 1: Comparison of word detection and insertion percentages of three selected parsers, with input from the original and high-quality lattices.

Table 1 compares the percentage of correct words and the percentage of word insertions for the three of the parsers using the original and high-quality lattices. The two parsers that make use of semantic knowledge perform significantly better than the allword parser. While the tags for the semantic network parser and the semantic trigram parser are identical, the semantic trigram parser obtains slightly greater recognition accuracy because it evaluates the likelihood of a sequence. The semantic network parser rejects illegal sequences of words but performs no reordering of legal ones. These results demonstrate that the parsers using trigrams with semantic knowledge can equal or better the performance of parsers that employ a finite-state grammar.

Effect of the Rank of Correct Words

Figure 2 shows the effects of reducing the rank of correct words when all correct words are in the lattice. As the quality of the lattices worsens, all parsers produce fewer correct words in their best hypotheses. The application of syntactic and semantic constraints produces improved accuracy, and the more specific the constraints, the greater the accuracy. The output of the allword parser is the most severely affected as the lattice quality worsens.

Effect of Missing Words

Parser outputs for sets of lattices with missing words are shown in Figure 3, and the results exhibit a different trend. When only 10 percent of the words are missing, the constrained parsers that use syntax or semantics produce greater word accuracy than the allword parser. This is because a significant number of sentences have no missing words and the constraints are useful in parsing these sentences. Since the average length of sentences in the email task is five words, roughly 60 percent of the sentences have no missing words when 10 percent of the correct words are missing from the word lattice. When 25 percent of the correct words are missing, only about 24 percent of the sentences should have no missing words. In this case, the more specific tags produce poor performance. The semantic trigram parser prooduces worse word accuracy than the allword parser, while the use of the more general tags still provides some benefit over the allword parser. The more specific tags are, the better they are able to differentiate between sequences of correct and incorrect words. Parsers with more specific tags are more disrupted by missing words, however, because there is less of a chance that other (incorrect) words that are present could produce an acceptable sequence of tags. Hence, the more general tags do not provide as much accuracy when all words are present, but they still may provide some benefit if many words are missing.

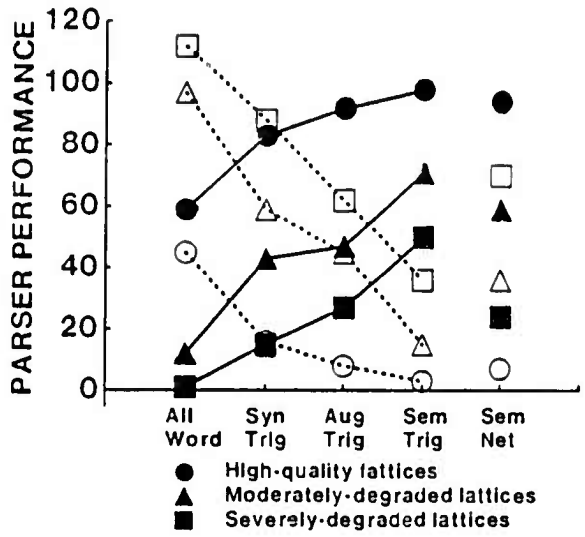


Figure 2: Effect of the rank of correct words on recognition accuracy. Filled symbols indicate correct word detection percentage; open symbols indicate insertion percentage. Parsers examined (from left to right) are the allword, syntactic trigram, augmented syntactic trigram, semantic trigram, and semantic network.

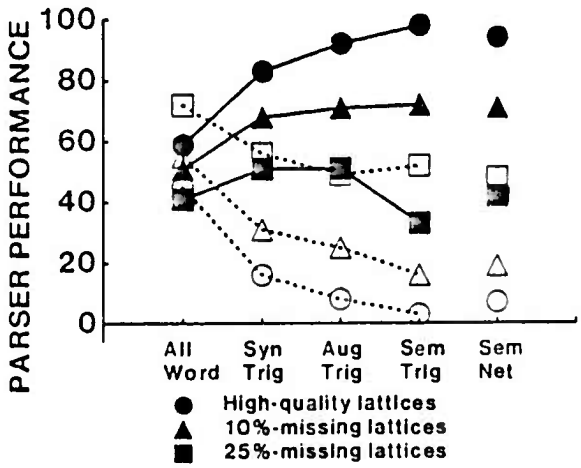


Figure 3: Effect of the rank of correct words on presence of missing words. Filled symbols indicate correct word detection percentage; open symbols indicate insertion percentage. Parser labels are as in Figure 2.

Effect of Syntax of Training Data

We also performed an additional experiment to examine the dependence of the parser that used the syntactic tags from the Brown corpus on the syntax of its training data. This was accomplished by estimating probabilities of the trigrams of the syntactic trigram parser using the following three different sets of sentences as the training text:

1. 171 examples of email sentences. (50 of these sentences were used as the test set in all experiments.)
2. 171 sentences from the original Brown corpus. (These were examples taken from articles in newspapers.)
3. 342 sentences obtained by combining the first two data sets.

The word accuracy of the syntactic trigram parsers trained on each of the above sets of example sentences is shown in Table 2. For all sets of lattices, parser performance increased as the training set more closely resembled the email sentences that the parsers were evaluated on.

It is not hard to imagine why the word accuracy was so low when the parsers were trained on only the Brown corpus sentences. Almost all of the Brown corpus sentences are declarative, and the first word tends to be an article, adjective, or noun. The email sentences, on the other hand, are all imperative or interrogative in form, and they begin with a verb, verb auxiliary, or *wh*-type adverb. Since the parsing proceeds in left-to-right fashion, the first word in the sentence has a great effect on how the rest of the sentence is parsed. In light of the profound differences between the syntactic forms of sentences in the Brown corpus and in the email task, the relatively good performance of the parser when trained on the combination of the two databases is quite encouraging. We believe this may indicate that reasonable performance may be obtained from a completely domain-independent syntactic parser, provided that all syntactic sentence forms are included in the training database.

	Brown Training	Mixed Training	Email Training
DETECTION PERCENTAGES:			
Original lattices	58	77	80
High-quality lattices	61	77	83
INSERTION PERCENTAGES:			
Original lattices	46	27	22
High-quality lattices	44	24	16

Table 2: Effect of the syntax of the training set of the syntactic trigram parser on word detection and insertion percentages.

SUMMARY

We compared the word recognition accuracy obtained using several different types of left-to-right sentence parsers. For the 325-word email task, we found that parsers using trigram representations of a small number of lexical or semantic tags could perform as well as or better than the parser using a finite-state grammar. Increasing the specificity of the trigram representation for a particular task domain tended to improve performance when the correct words are not among the very best word candidates, but it can degrade performance if correct words are missing completely from the input word lattices. The performance of the syntactic trigram parser appeared to be relatively insensitive to the specific contents of its training database, provided that the training set included the sentence forms that were encountered in the test sentences.

REFERENCES

1. Lowerre, B., and Reddy, D. R., *The Harpy Speech Understanding*, in Trends in Speech Recognition, W. A. Lea, ed., Lawrence Erlbaum, New York, 1979.

2. Erman, L. D., and Lesser, V. R., *The Hearsay-II Speech Understanding System: A Tutorial*, in Trends in Speech Recognition, W. A. Lea, ed., Lawrence Erlbaum, New York, 1979.

3. Kimball, O., Price, P., Roucos, S., Schwartz, R., Kubala, F., Chow, Y.-L., Haas, A., Krasner, M., Makhoul, J., "Recognition Performance and Grammatical Constraints", *Proceedings of the DARPA Speech Recognition Workshop*, Science Applications International Corporation Report Number SAIC-86/1546, 1986, pp. 53-59.

4. Bahl, L. R., Jelinek, F., and Mercer, R., L., "A Maximum Likelihood Approach to Continuous Speech Recognition", *IEEE Trans. on Patt. Anal. and Mach. Intell.*, Vol. 51983.

5. Cole, R., Phillips, M., Brennan, B., Chigier, B., "The C-MU Phonetic Classification Sysytem", *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1986, pp. 2255-2257.

6. Rudnicky, A. I., "The Lexical Access Component of the CMU Continuous Speech Recognition System", *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1987.

7. Hayes, P. J., Hauptmann, A. G., Carbonell, J. G., and Tomita, M., "Parsing Spoken Language: A Semantic Caseframe Approach", *Proceedings of COLING-86*, 1986.

8. Francis, W. M., and Kucera, K. K., *A Standard Corpus of Present-Day Edited American English, for Use with Digital Computation*, Brown University Department of Linguistics, Providence RI, 1964.

9. Hayes, P. J., "Entity-Oriented Parsing", *Proceedings of COLING-84*, 1984.

[1]This research was sponsored in part by the National Science Foundation, Grant IRI-85-12695, and in part by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under Contract N0039-85-C-0163. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

Presented at the IEEE International Conference on Acoustics, Speech, and Signal Processing, Dallas, Texas, April 6-9, 1987.

A Stochastic Segment Model for Phoneme-Based Continuous Speech Recognition

S. Roucos and M. O. Dunham

BBN Laboratories Incorporated
Cambridge, MA 02238

Abstract -- Developing accurate and robust phonetic models for the different speech sounds is a major challenge for high performance continuous speech recognition. In this paper, we introduce a new approach, called the stochastic segment model, for modelling a variable-length phonetic segment X, an L-long sequence of feature vectors. The stochastic segment model consists of 1) time-warping the variable-length segment X into a fixed-length segment Y called a resampled segment, and 2) a joint density function of the parameters of the resampled segment Y, which in this work is assumed Gaussian. In this paper, we describe the stochastic segment model, the recognition algorithm, and the iterative training algorithm for estimating segment models from continuous speech. For speaker-dependent continuous speech recognition, the segment model reduces the word error rate by one third over a hidden Markov phonetic model.

1. Introduction

In large vocabulary speech recognition, a word is frequently modelled as a network of phonetic models. That is, the word is modelled acoustically by concatenating phonetic acoustic models according to a pronunciation network stored in a dictionary of phonetic spellings. In phoneme-based speech recognition systems, it is not necessary for the speaker to train all words in the vocabulary; only the phonetic models are trained. Assuming the above structure for a speech recognition system, the goal of this work is to look for an improved approach to phonetic modelling.

Hidden Markov modelling (HMM) is one method for probabilistic modelling of the acoustic realization of a phoneme. Although the HMM approach has been used successfully [1, 2, 3], its recognition performance is not sufficiently accurate for large vocabulary continuous speech recognition. We propose an alternative and novel approach, called a stochastic segment model, with the goal of improving phonetic modelling. The motivation for looking at speech on a segmental level, rather than on a frame-by-frame basis as in HMM or dynamic time warping (DTW), is that we can better capture the spectral/temporal relationship over the duration of a phoneme. Evidence of the importance of spectral correlation over the duration of a segment can be found in the success of segment-based vocoding systems [4].

A speech "segment" is a variable-length sequence of feature vectors, where the features might be, for example, cepstral coefficients. The stochastic segment model is defined on a fixed-length representation of the observed segment, which is obtained by a time-warping (or resampling) transformation. The stochastic segment model is a multivariate Gaussian density function for the resampled representation of a segment. The recognition algorithm chooses the phoneme sequence that maximizes a match score on the resampled segments. The training algorithm iterates between two steps: first, the maximum probability phonetic segmentation of the input speech is obtained, then maximum likelihood density estimates of the segment models are derived.

The paper is organized as follows. Section 2 introduces the segment model. Section 3 describes the segment-based recognition algorithm, and Section 4 describes the training algorithm. Section 5 presents experimental results for phoneme and word recognition, comparing the results to HMM recognition results for the same tasks. Finally, Section 6 contains a brief summary.

2. Stochastic Segment Model

In this section, we define the stochastic segment model for an observed sequence of speech frames $X = [x_1\ x_2 \ldots x_L]$, where x_l is a k-dimensional feature vector. We can think of this observation as a variable-length realization of an underlying fixed-length spectral trajectory $Y = [y_1\ y_2 \ldots y_m]$ where the duration of X is variable due to variation in speaking rate. Given X, we define the fixed-length representation $Y = XT_L$ where the $L \times m$ matrix T_L, called the resampling transformation, represents a time-warping. The segment Y, called a *resampled* segment, is an m-long sequence of k-dimensional vectors (or a $k \times m$ matrix). The stochastic segment model for each phoneme α is based on the resampled segment Y and is a conditional probability density function $p(Y|\alpha)$. The density $p(Y|\alpha)$ is assumed to be multivariate Gaussian which is a km-dimensional model for the entire fixed-length segment Y.

Resampling Transformations

The resampling transformation T_L is an $L \times m$ matrix used to transform an L-length observed segment X into an m-length resampled segment Y. We considered several different variable- to fixed-length transformations, concentrating on transformations which had previously been evaluated in the segment vocoder [4]. The best recognition results are obtained using linear time sampling without interpolation. Linear time sampling involves choosing m uniformly spaced times at which

to sample the segment trajectory. Sampling without interpolation refers to choosing the nearest observation in time to the sample point, rather than interpolating to find a value at the sample point.

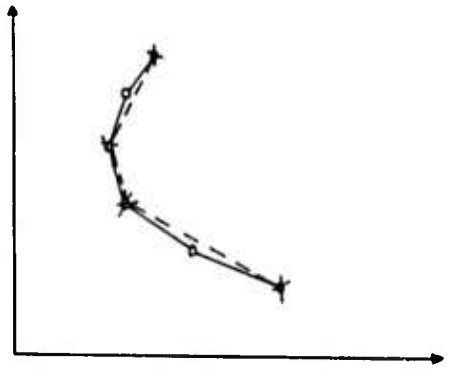


Figure 1: Input segment (o) and corresponding resampled segment (x). The two axes correspond to two cepstral coefficients.

Figure 1 shows an input segment with duration six in two-dimensional space and the corresponding resampled Y (with m = 4) using linear time warping without interpolation. The resampling transformation in this case is:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Probabilistic Model

As already mentioned, the segment model is a multivariate Gaussian based on the resampled segment Y, $p(Y|\alpha)$. Recall that resampled segments are km-dimensional, where k is the number of spectral features per sample and m is the number of samples. In this work, typically k=14 and m=10. Consequently, the segment model has 140 dimensions. Because of insufficient training, we cannot estimate the full phoneme-dependent covariance matrix, so we must make some simplifying assumptions about the structure of the problem. For the experiments reported here, we assume that the m samples of the resampled segment are independent of each other, which gives a block diagonal covariance structure for Y, where each block in the segment covariance matrix corresponds to the $k \times k$ covariance of a sample. The log of the conditional probability of a segment Y given phoneme α can then be expressed as

$$ln[p(Y|\alpha)] = \sum_{j=1}^{m} ln[p_j(y_j|\alpha)], \qquad (1)$$

where $p_j(y_j|\alpha)$ is a k-dimensional multivariate Gaussian model for the j-th sample in the segment. The block-diagonal structure saves a factor of m in storage and a factor of m^2 in computation. The disadvantage of this approach is that the assumption of independence is not valid, particularly if resampling does not use interpolation where adjacent samples may be identical. In the future, with more training data, we hope to relax this assumption. It is likely that more detailed probabilistic models, such as Gaussian mixture models [5] and context-dependent (conditional) models [2, 3], will yield better recognition results than the simple Gaussian model. However, due to larger training requirements we did not pursue these models in this work.

Properties of the Segment Model

There are several aspects of the stochastic segment model which are useful properties for a speech recognition system. First, the transformation T_L, which maps the variable-length observation to a fixed-length segment, can be designed to constrain the temporal structure of a phoneme model so that all portions of the model are used in the recognition. We conjecture that the fixed transformation will provide a better model of phoneme temporal/spectral structure than either HMM or DTW. Second, the segment model is a joint representation of the phoneme, so the model can capture correlation structure on a segmental level. In HMM, frames are assumed independent given the state sequence. In the segment model, no assumptions of independence are *necessary*, though the model of Y given by Equation 1 is based on the assumption of sample independence because of limited training data in this study. The model is potentially more general than the special case of (1). Lastly, by using a segment model we can compute segment level features for phoneme recognition. In other words, the segment model provides a good structure for incorporating acoustic-phonetic features in a statistical (rather than rule-based) recognition system. For example, one might want to measure and incorporate formant frequency or energy differences over a segment. Section 5 includes results where sample duration is used as a feature, which can only be computed given the length of the entire segment.

3. Recognition Algorithm

In this section, we describe the recognition algorithm. First, we describeconsider the case when the input is phonetically hand-segmented. Then, we generalize to automatic recognition, that is, joint segmentation and recognition of continuous speech.

When the segmentation of the input is known, we consider a single segment X independently of neighboring segments. The input segment X is resampled as segment Y. The recognition algorithm is then to find the phoneme $\hat{\alpha}$ that maximizes $p(Y|\alpha)$:

$$\hat{\alpha} = \arg\max_{\alpha} ln[p(Y|\alpha)p(\alpha)] \qquad (2)$$

where $\ln[p(Y|\alpha)]$ is given by Equation 1. This decision rule is equivalent to a maximum a-posteriori rule.

In an automatic recognition system, it is necessary to find the segmentation as well as to recognize the phonemes. In this case, we hypothesize all possible segmentations of the input, and for each hypothesized segmentation $\underline{s}$ of the input into n

segments, we choose the sequence of phonemes $\hat{\underline{\alpha}}$ that maximizes:

$$J(\underline{s}) = \sum_{i=1}^{n} \{ L(i) \ln[p(Y_i|\hat{\alpha}_i)p(\hat{\alpha}_i)] + C \} \quad (3)$$

where L(i) is the duration of the i-th segment, Y_i is the resampled segment corresponding to the i-th segment in $\underline{s}$, and $\hat{\alpha}_i$ is the phoneme that maximizes $p(Y_i|\alpha)p(\alpha)$. The cost C is adjusted to control the segment rate. An efficient solution to joint segmentation and recognition is implemented using a dynamic programming algorithm. Note that for joint segmentation and recognition, it is necessary to weight the segment probability by the duration of the segment, so that longer segments contribute proportionately higher scores to the match score J(.) of the whole sequence.

4. Training Algorithm

In this section, we present the training algorithm for estimating the segment models from continuous speech. We assume that the phonetic transcription of the training data is known and that we have an initial Gaussian model, $p_0(Y|\alpha)$ for all phonemes. (Phonetic transcriptions can be generated automatically from the word sequence that corresponds to the speech by using a word pronunciation dictionary.) We assume that the phonetic sequence $\underline{\alpha}$ has length n. The algorithm comprises two steps: automatic segmentation and parameter estimation. The algorithm maximizes the log likelihood of the optimal segmentation for the phonetic transcription, where the log likelihood of a segmentation $\underline{s}$ is given by:

$$l(\underline{s}) = \sum_{i=1}^{N} \ln[p(Y_i|\alpha_i)p(\alpha_i)] \quad (4)$$

where Y_i is the resampled segment that corresponds to the i-th segment in the segmentation $\underline{s}$ and α_i is the i-th phoneme in the sequence $\underline{\alpha}$. With t = 0, the iterative algorithm is given by:

1. Find the segmentation $\underline{s}_t$ of the training data that maximizes $l(\underline{s}_t)$ for the given transcription and the current probability densities $\{p_t(Y|\alpha)\}$.
2. Find the maximum likelihood estimate for the densities $\{p_{t+1}(Y|\alpha)\}$ of all phonemes, using the segmentation $\underline{s}_t$.
3. t <- t + 1 and go to Step 1

Both steps of the algorithm are guaranteed to increase $l(\underline{s}_t)$ with t. If there are at least two *different* observations of every phoneme, then the probability of the sequence is bounded. Hence, the iterative training algorithm converges to a local optimum. Step 1 is implemented as a dynamic programming search whose complexity is linear with the number of phonetic models N. Step 2 is the usual sample mean and sample covariance maximum likelihood estimates for Gaussian densities.

5. Experimental Results

In this section we will present results for a phoneme recognition task, as well as word recognition results for a segment-based recognition system and an HMM-based system. All experiments use $m = 10$ samples per segment and $k = 14$ mel-frequency cepstral coefficients per sample. These values are based on work in segment quantization [6], and limited experimentation confirmed that these values represent a reasonable compromise between complexity and performance. Speech is sampled at 20 kHz, and analyzed every 10 ms with a 20 ms Hamming window.

Phoneme Recognition

The database used for phoneme recognition is approximately five minutes of continuous speech from a single speaker. The test set contains 270 phonemes. Both the test set and the training set are hand-labelled and segmented, using a 61 symbol phonetic alphabet. In counting errors, an 'AX' (schwa) recognized as 'IX' (fronted schwa) is considered acceptably correct, as is an 'URT' (unreleased T) recognized as a 'T'. All recognition rates presented represent "acceptably correct" recognition rates. The acceptable recognition rate is typically 6% to 8% higher than the strictly correct recognition rate.

Phoneme recognition results for three different cases are given in Table 1. The results illustrate a small degradation in performance due to moving from recognition based on manually segmented data to automatic recognition. Using automatic training does not degrade performance any further.

We also experimented with using an additional segmental feature to the cepstral parameters: sample duration which requires knowledge of the hypothesized duration of the segment. Using joint segmentation and recognition with hand-segmented training data, performance improved from 74.4% to 75.9% as a result of using the duration feature.

Training Segmentation	Test Segmentation	% Recognition	% Insertion
Manual	Manual	78.5	0.0
Manual	Automatic	74.4	10.0
Automatic	Automatic	73.7	7.8

Table 1: Recognition results using manually segmented speech and automatically segmented speech.

For reference, a discrete hidden Markov model with 3 states/phoneme and using a codebook with 256 entries has 62% phonetic recognition rate with 12% insertions. The HMM recognition performance on this database is higher when phoneme models are conditioned on left context, 75% correct with 12% insertions [2]. In the latter case, 600 left-context phonetic models are used in the HMM system while 61 phonetic models are used in the stochastic segment model.

Word Recognition

The segment-based word recognition system consists of a dictionary of phoneme pronunciation networks and a collection of segment phoneme models. A word model is built by

concatenating phoneme models according to the pronunciation network. The recognition algorithm is simply a dynamic programming search (Viterbi decoding) of all possible word sequences. For the results in this paper, we assume that words are independent and equally probable; there is no grammar (statistical or deterministic) associated with recognition. Within each word, we find the best phoneme segmentation for that word, where the phoneme sequence is constrained by the word pronunciation network.

For continuous speech word recognition, we used a 350 word vocabulary, speaker-dependent database based on an electronic mail task. We present results for three different male speakers. Fifteen minutes of speech was used for training the 61 phoneme models for each speaker, from which the word models were then built. An additional 30 sentences (187 words) are used for recognition. Analysis parameters are the same as for the previous database. Again, "acceptable" error rates are reported here, where in this case, homophones such as "two" and "to" are considered acceptable errors. Since we do not use a grammar, homophones are indistinguishable.

The initial segment models are obtained on training from segmentations given by a discrete hidden Markov model recognition system. The results after one pass of training of the segment model for the three speakers are summarized in Table 2. The HMM recognition results are also given for comparison. For the HMM results, five passes of the forward-backward training algorithm are performed. The segment phoneme system outperforms the phoneme-based HMM system, reducing the error rate by one third (including insertions). However, the segment phoneme system does not quite match the HMM context model system. This suggests that context-dependent segment models might be useful. Note that in the earlier phoneme results, the segment system matched the performance of HMM models conditioned on left context only. Here we give results for HMM models conditioned on both left and right context. The HMM system with context models conditioned on both left and right context uses 2000 models, or thirty times the number used by the segment system.

Speaker	Segment-PH	HMM-PH	HMM-PH-LE-RI
RS	87/5.3	85/10.2	90/1.1
FK	83/2.1	75/ 5.4	88/2.7
AW	78/3.7	68/ 7.5	86/3.7
Average	83/3.7	76/ 7.7	88/2.5

Table 2: Word recognition/insertion rates for three speakers for the segment phoneme system and for two HMM systems: phoneme models and phoneme models conditioned on the left and right context.

6. Conclusion

To summarize, we feel that the segment model offers the potential for large improvements in speaker-dependent acoustic modelling of phonemes in continuous speech. Our initial results demonstrate the potential of the approach. Of course, a practical system requires automatic training and recognition, which we demonstrated to perform close to the hand-segmented case at the cost of a few insertions. For comparison, the automatic segment system reduces the word error rate by one third over an HMM system on a 350-word continuous speech recognition task.

Acknowledgements

This research was supported by the Advanced Research Projects Agency of the Department of the Defense and was monitored by ONR under Contract No. N00014-85-C-0279.

References

1. L.R. Bahl, F. Jelinek, and R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 2, March 1983, pp. 179-190.

2. R.M. Schwartz, Y.L. Chow, O.A. Kimball, S. Roucos, M. Krasner, and J. Makhoul, "Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Tampa, FL, March 1985, pp. 1205-1208, Paper No. 31.3.

3. Y.L. Chow, R.M. Schwartz, S. Roucos, O.A. Kimball, P.J. Price, G.F. Kubala, M.O. Dunham, M.A. Krasner, and J. Makhoul, "The Role of Word-Dependent Coarticulatory Effects in a Phoneme-Based Speech Recognition System", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Tokyo, Japan, April 1986, pp. 1593-1596, Paper No. 30.9.1.

4. S. Roucos, R. Schwartz, and J. Makhoul, "Segment Quantization for Very-Low-Rate Speech Coding", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Paris, France, May 1982, pp. 1565-1569.

5. B. -H. Juang and L.R. Rabiner, "Mixture Autoregressive Hidden Markov Models for Speech Signals", *IEEE Trans. Acoust., Speech and Signal Proc.*, Vol. ASSP-33, No. 6, December 1985, pp. 1404-1413.

6. S. Roucos, R. Schwartz, and J. Makhoul, "A Segment Vocoder at 150 B/S", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Boston, MA, April 1983, pp. 61-64.

Presented at the IEEE International Conference on Acoustics, Speech, and Signal Processing, Dallas, Texas, April 6-9, 1987.

Rapid Speaker Adaptation using a Probabilistic Spectral Mapping

Richard Schwartz, Yen-Lu Chow, Francis Kubala
BBN Laboratories
10 Moulton St.
Cambridge, MA 02238

Abstract

This paper deals with rapid speaker adaptation for speech recognition. We introduce a new algorithm that transforms hidden Markov models of speech derived from one "prototype" speaker so that they model the speech of a new speaker. The speaker normalization is accomplished by a probabilistic spectral mapping from one speaker to another. For a 350 word task with a grammar and using only 15 seconds of speech for normalization, the recognition accuracy is 97% averaged over 6 speakers. This accuracy would normally require over 5 minutes of speaker dependent training. We derive the probabilistic spectral transformation of HMMs, describe an algorithm to estimate the transformation, and present recognition results.

1. Introduction

We have previously demonstrated our techniques for robust modeling of phonetic coarticulation for large vocabulary, continuous speech recognition [1]. The technique combines detailed *context-dependent* phonetic hidden Markov models (HMM) with robust *context-independent* models to improve word recognition accuracy. The BBN Speech Recognition System (BYBLOS) integrates many components to allow accurate speech recognition with a grammar [2]. On a 350-word continuous speech recognition task, the word recognition accuracy was 90% with no grammar, and 98%-99% with a grammar. To achieve this high recognition accuracy, each speaker read 300 training sentences or about 15 minutes of training speech.

Some speech recognition applications have a need for a new speaker to begin using the system with reasonable accuracy without investing a long time to train the system on their voice. However, as we will see in section 4, the speaker-dependent performance degrades dramatically when the amount of training speech is reduced using the standard training procedure.

The approach that we consider in this paper is to normalize well-trained models from a "prototype" speaker, to model the speech of the new speaker. The normalization requires only a few sentences (referred to as "normalization speech") from the new speaker.

In Section 2, we derive and present a procedure for estimating a probabilistic spectral mapping from one speaker to another. Experiments to test these procedures are described in Section 3. The results of the experiments are analyzed in Section 4.

2. Probabilistic Mapping

In this section we present the basis for the probabilistic transformation and show it to be equivalent to an expanded HMM model for each state of the original HMM. The transformation is generalized to be partially dependent on the particular phoneme. Finally, we present two detailed algorithms for estimating the probabilistic mapping.

Discrete Hidden Markov Models

For each state of a discrete HMM, we have a discrete probability density function (pdf) defined over a fixed set, N, of spectral templates. For example, in the BYBLOS system we typically use a vector quantization (VQ) codebook of size N=256 [3]. The index of the closest template is referred to below as the "quantized spectrum". We can view the discrete pdf for each state s as a probability row vector

$$\underline{p}(s) = [p(k_1|s),\ p(k_2|s),\ \ldots,\ p(k_N|s)], \qquad (1)$$

where $p(k_i|s)$ is the probability of spectral template k_i at state s.

Mapping From Prototype to New Speaker

If we define a quantized spectrum for the prototype speaker as k_i, $1 \le i \le N$, where i is the index of the spectral template and a quantized spectrum for the new speaker as k'_j, $1 \le j \le N$, then we denote the probability that the new speaker will produce quantized spectrum k'_j, given that the prototype speaker produced spectrum k_i as $p(k'_j|k_i)$ for all i, j.

We can rewrite the probability for spectrum k'_j given a particular state s of the HMM as

$$p(k'_j|s) = \sum_{i=1}^{N} p(k_i|s)\ p(k'_j|k_i,s) \qquad (2)$$

If we assume that the probability of k' given k is independent of s, then

$$p(k'_j|s) = \sum_{i=1}^{N} p(k_i|s)\ p(k'_j|k_i) \qquad (3)$$

The set of probabilities $p(k'_j|k_i)$ for all i and j form an $N \times N$ matrix, $\mathbf{T}$ that can be thought of as a probabilistic transformation from one speaker's spectral space to another's. We can compute the discrete pdf, $\underline{p}'(s)$ at state s for the new speaker as the product of the row vector, $\underline{p}(s)$ and the matrix, $\mathbf{T}$.

$$\underline{p}'(s) = \underline{p}(s)\ \mathbf{T}; \qquad \mathbf{T}_{ij} = p(k'_j|k_i) \qquad (4)$$

Expanded HMM Formulation

The probabilistic transformation can also be described in terms of an expanded HMM model for the state. Figure 1a shows a single state of the HMM for a new speaker. It contains a single discrete probability vector, $\underline{p}'(s)$. Figure 1b shows an expanded model in which the single state is replace by N parallel paths.

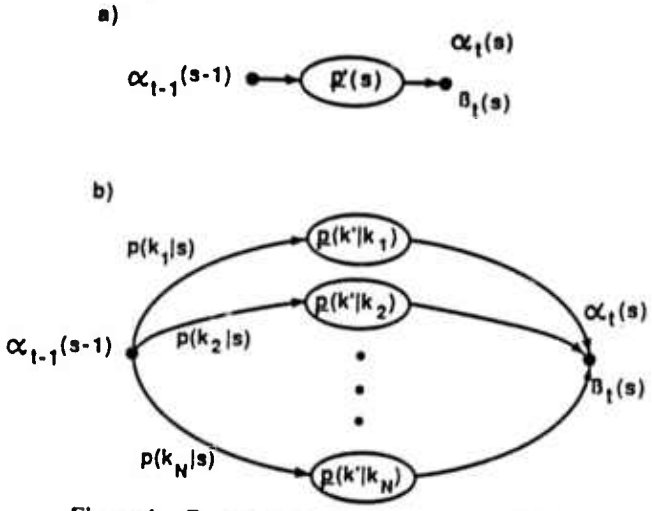


Figure 1: Expanded HMM. a) single state of the HMM; b) expanded model separating prototype pdf and transformation matrix.

The transition probability for path i is $p(k_i|s)$, the probability of the quantized spectrum, k_i given the same state s for the prototype speaker. The discrete pdf on that path is $\underline{p}(k'|k_i)$, which corresponds to row i of the transformation matrix.

Careful inspection of the figure will reveal that the probability of any new-speaker spectrum, k'_j for the expanded HMM shown is a summation of the jth probability over all N paths, as given in (3). Therefore, Figure 1a represents the left side of equation 4, while figure 1b represents the right side. Now that we have decomposed each pdf for the new speaker into its components, we can use the forward-backward algorithm to estimate the transformation matrix while keeping the prototype pdf fixed. Then, once the matrix has been determined, we can replace the expanded HMM by the single pdf resulting from the vector-matrix multiplication in (4).

Phoneme-Dependent Transformation

The independence assumption in (3) above assumes that a single (probabilistic) spectral mapping will transform the speech of one speaker to that of another. However, we know that some of the differences between speakers cannot be modeled this simply. We can define a phoneme-dependent mapping:

$$p(k'_j|s) = \sum_{i=1}^{N} p(k_i|s)\ p(k'_j|k_i,\phi(s)) \qquad (5)$$

where $\phi(s)$ specifies the equivalence class of states in models that represent the same phoneme as s. Since the amount of training speech from the new speaker will be small, we could not hope to have enough samples of each phoneme to estimate a reliable mapping for all phonemes. Therefore, we interpolate the *phoneme-dependent* transformation matrix with the *phoneme-independent* transformation matrix. The weight for the combination depends on the number of observed frames of the particular phoneme. Thus for those phonemes that occur several times in the normalization speech, the transformation will depend mostly on that particular phoneme.

Detailed Algorithm

The algorithm begins with a VQ codebook and well-trained *context-dependent* and *context-independent* pdfs derived from a prototype speaker. A small number of sentences are read by the new speaker. The new (normalization) speech is quantized using the prototype speaker's VQ codebook. (This step may be a source of reduced performance, and will be discussed further in Section 4.) Then, we use a modification of the standard forward-backward algorithm to estimate the *phoneme-dependent* and *phoneme-independent* transformation matrices.

To save computation and storage we use $\underline{p}'(s)$, the compact HMM in Figure 1a to compute the partial (α and β) terms in the forward-backward algorithm. The forward backward "counts" are added to a separate count matrix. (Two methods for computing the counts are defined at the end of this subsection.) Since we have no *a priori* transformation matrix, we must provide an initial estimate. To minimize computation we use an identity matrix for the first transformation (that is, we just use the prototype pdf as is). However, when we compute the counts in the first pass, the transformation matrix is a constant value of $1/N$. After the first pass, the same matrix is used both for forward-backward partial terms and for computing the counts. At the end of each pass through the normalization data, each row of the count matrix, which corresponds to $\underline{p}(k'|k_i)$, the transformation given one prototype spectrum k_i is rescaled so it sums to 1. This normalized count matrix then becomes the new probabilistic transformation matrix. After the final pass we transform all the prototype models using (4).

Computing Counts - Method 1:

For each alignment of a state with an observed quantized spectrum, $k'(t)=k'_j$, the prototype pdf vector, $\underline{p}(s)$, is multiplied by column j of the transformation matrix, $p(k'_j|k_i)$ $1 \le i \le N$. This vector product is multiplied by the constants $\alpha_{t-1}(s-1)$ and $\beta_t(s)$ (shown in Figure 1b) and then accumulated in column j of the count matrix. $\alpha_{t-1}(s-1)$ is the probability of the observed spectra from frames 1 through $t-1$ given the models up to but not including state s. $\beta_t(s)$ is the probability of the observed spectra from the end of the sentence back to time $t+1$ given the models after state s. This method corresponds to the standard (maximum likelihood) forward-backward algorithm for the HMM shown in Figure 1b.

Computing Counts - Method 2:

Method 2 is similar to Method 1, with the exception that the prototype pdf vector is multiplied by the constants $\alpha_t(s)$ and $\beta_t(s)$ (shown in Figure 1b) and then added to the corresponding column of the count matrix. That is the counts are computed as the probability of being in state s at time t, times the prototype

pdf. We found that only one pass of the algorithm is necessary for Method 2, making it preferable in terms of computation. We also found that this method results in slightly better performance than Method 1. Therefore all results quoted below are for Method 2.

3. Experiments

Database

We have performed experiments on a 350-word subset of a naval database retrieval task (FCCBMP). The task has a fairly rich structure and allows many different types of questions and commands. The prototype speaker recorded 400 sentences in 4 sessions of 100 sentences each, separated by a few days. The first three sessions were designated as training data, and the last as test material. At an average of 3 seconds per sentence, the total duration of the training material was thus about 15 minutes for the prototype speaker.

Each of 6 new speakers then recorded a subset of the training sentences and, in a separate session, the 100 test sentences. The 6 speakers included one female, one non-native speaker, one experienced speaker, and three inexperienced speakers.

We constructed a dictionary of phonetic pronunciations for the vocabulary without listening to either the training or test material. With very few exceptions, only one pronunciation was chosen for each word.

The sentences were read directly into a close-talking microphone in a natural but deliberate style in a quiet office environment. The speech was lowpass filtered at 10 kHz and sampled at 20 kHz. Fourteen Mel-frequency cepstral coefficients (MFCC) were computed every 10 ms on a 20 ms analysis window. One half of the training speech of the prototype speaker was used to derive a speaker-dependent VQ codebook. Then all the recorded speech for all speakers was quantized using this codebook.

Training

The 15 minutes of speech from the prototype speaker was used, together with the phonetic dictionary to estimate *context-dependent* and *context-independent* phonetic models. The speech models for the new test speakers were computed in two ways: Speaker-Dependent training and Speaker Normalization. In addition to these two models for the new speaker, we also performed control experiments using the prototype speaker's models without any change. These unaltered models are designated "Cross-Speaker" models. Prior to recognition, the phonetic models were combined and concatenated into word models to facilitate the word recognition process.

Recognition

We used the time-synchronous search procedure described in [4] to find the most likely sequence of words for each test sentence. Recognition experiments were performed both with and without a grammar. When no grammar was used, the effective branching factor was equal to the vocabulary size (350). The grammar used had a Maximum Perplexity [5] of 30 and an estimated Perplexity [6] of 20 (measured on a test set). The recognized sequence of words was then compared automatically to the correct answer to determine the percentage of errors of each type: substitutions, deletions and insertions.

4. Results

We use an error measure that reflects all three types of errors in a single number. The percent error is given by

$$\%error = 100\ \frac{\text{substitutions} + \text{deletions} + \text{insertions}}{\text{total words} + \text{insertions}}$$

The word accuracy is then defined as 100 - *%error*. Note that this definition is different from the percent correct words.

Figure 2 below shows the recognition error as a function of the amount of training speech (on a log scale) for both training conditions. For reference, the results using the Cross-Speaker models are also shown. Some of the conditions that did not seem to warrant extensive testing (e.g., 15 second speaker-dependent training) were evaluated using a subset of the speakers. More critical results (e.g., 15 second speaker normalization) were evaluated using all 6 speakers.

The recognition error varied less with the duration of speech for speaker normalization than for speaker-dependent training - particularly when a grammar was used. The error rate with 15 seconds of normalization speech was about the same as achieved by the speaker-dependent training method with 6 to 10 minutes of training speech. In particular, when a grammar was used, the word recognition error with only 15 seconds of normalization speech from each speaker was 4% (97% correct words with 1% insertions.)

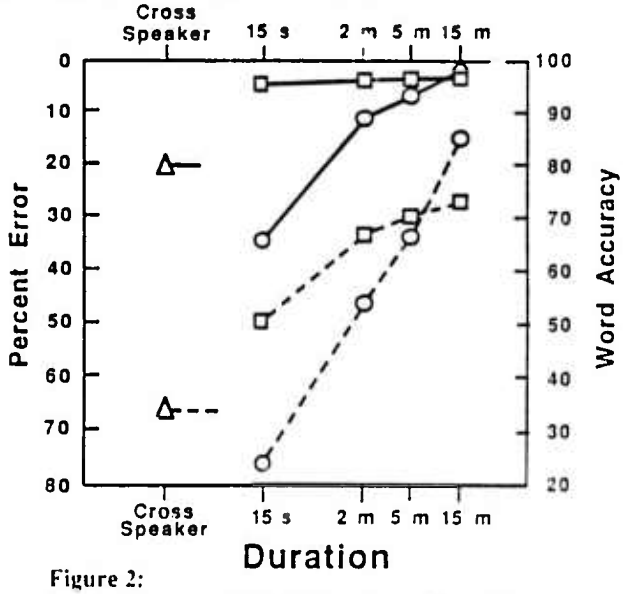


Figure 2:
Speaker-Dependent Training vs Normalization.
Speaker-Dependent Training (○); Speaker Normalization (□); Cross-Speaker Results (Δ). The solid line indicates accuracy with a grammar; the dashed line indicates no grammar.

Detail vs Robustness

We can see from the results with and without a grammar that the speaker transformation seems to be much more successful when a grammar is used. That is, the error decreased by a bigger factor (from speaker-dependent training to the normalization algorithm) when a grammar was used than when no grammar was used.

When no grammar is used in speech recognition it is important that the models be sharply tuned to make fine distinctions. Occassional errors will result from a finely tuned model that was inadequately trained. In contrast, we assume that when a grammar is used the number of words allowed at each point is small relative to the vocabulary size. In this case it is less likely that fine phonetic distinctions will be necessary. To get very high performance, it becomes more important that the correct word *never* get a very low score.

We have observed that the pdfs resulting from the speaker normalization procedure are typically broader than those resulting from speaker-dependent training. We surmise that this effect, combined with the appropriate spectral mapping between the speakers, accounts for the large improvement in accuracy when a grammar is used.

Source of Errors

We performed a series of experiments on one speaker in an effort to determine whether the major source of errors is the duration of normalization of speech, the normalization procedure itself, or the fact that the VQ codebook of the prototype speaker is used for the new speaker. We present the recognition results (using no grammar) in Table 1 below.

Condition	*% error*
15 min spkr-dependent training	16%
Prototype VQ codebook	24%
15 min normalization	27%
5 min normalization	30%
2 min normalization	33%

Table 1: Source of Recognition Errors.
Each line changes one experimental condition.

As we see in the table, the largest increase in word error is the result of using a VQ codebook that was not designed for the new speaker. Our next step, therefore, will be to derive a codebook for the new speaker from a combination of the new speech and the prototype speaker's codebook. This expanded codebook will form the basis for the normalized pdf models.

5. Summary

We have presented a method for transforming the discrete HMM models of one speaker so that they are appropriate for a second speaker. The procedure uses a small amount of speech to estimate a probabilistic spectral mapping from a well-trained prototype speaker to a new speaker. The recognition accuracy with 15 seconds of normalization speech and a grammar (tested on a set of 6 diverse speakers) was 97% with 1% word insertions.

The method also makes the HMM models more robust, which is most appropriate when a grammar is used. There is some evidence that the speaker normalization performance suffers because we use the prototype speaker's VQ codebook for the new speaker. In future work we will investigate speaker-adaptive VQ codebooks for speaker normalization.

Acknowledgement

This work was supported by the Defense Advanced Research Projects Agency and was monitored by the Office of Naval Research under contract number N00014-85-C-0279.

References

1. Y.L. Chow, R.M. Schwartz, S. Roucos, O.A. Kimball, P.J. Price, G.F. Kubala, M.O. Dunham, M.A. Krasner, and J. Makhoul, "The Role of Word-Dependent Coarticulatory Effects in a Phoneme-Based Speech Recognition System", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Tokyo, Japan, April 1986, pp. 1593-1596, Paper No. 30.9.1.

2. Y.L. Chow, M.O. Dunham, O.A. Kimball, M.A. Krasner, G.F. Kubala, J. Makhoul, P.J. Price, S. Roucos, and R.M. Schwartz, "BYBLOS: The BBN Continuous Speech Recognition System", *IEEE Int Conf. Acoust., Speech, Signal Processing*, Dallas, TX, April 1987, Elsewhere in these proceedings

3. J. Makhoul, S. Roucos, and H. Gish, "Vector Quantization in Speech Coding", *Proc. IEEE*, Vol. 73, No. 11, November 1985, pp. 1551-1588, Special Issue on Man-Machine Speech Communication.

4. R.M. Schwartz, Y.L. Chow, O.A. Kimball, S. Roucos, M. Krasner, and J. Makhoul, "Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Tampa, FL, March 1985, pp. 1205-1208, Paper No. 31.3.

5. M.M. Sondhi and S.E. Levinson, "Computing Relative Redundancy to Measure Grammatical Constraint in Speech Recognition Tasks", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Tulsa, OK, April 1978, pp 409-412.

6. L.R. Bahl, F. Jelinek, and R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 2, March 1983, pp. 179-190.

Presented at the IEEE International Conference on Acoustics, Speech, and Signal Processing, Dallas, Texas, April 6-9, 1987.

BYBLOS: The BBN Continuous Speech Recognition System

Y.L. Chow, M.O. Dunham, O.A. Kimball, M.A. Krasner,
G.F. Kubala, J. Makhoul, P.J. Price, S. Roucos,
and R.M. Schwartz

BBN Laboratories Incorporated
10 Moulton Street
Cambridge, MA 02239

Abstract

In this paper, we describe BYBLOS, the BBN continuous speech recognition system. The system, designed for large vocabulary applications, integrates acoustic, phonetic, lexical, and linguistic knowledge sources to achieve high recognition performance. The basic approach, as described in previous papers [1, 2], makes extensive use of robust context-dependent models of phonetic coarticulation using Hidden Markov Models (HMM). We describe the components of the BYBLOS system, including: signal processing frontend, dictionary, phonetic model training system, word model generator, grammar and decoder. In recognition experiments, we demonstrate consistently high word recognition performance on continuous speech across: speakers, task domains, and grammars of varying complexity. In speaker-dependent mode, where 15 minutes of speech is required for training to a speaker, 98.5% word accuracy has been achieved in continuous speech for a 350-word task, using grammars with perplexity ranging from 30 to 60. With only 15 seconds of training speech we demonstrate performance of 97% using a grammar.

1. Introduction

Speech is a natural and convenient form of communication between man and machine. The speech signal, however, is inherently variable and highly encoded. Vast differences occur in the realizations of speech units related to context, style of speech, dialect, talker. This makes the task of large vocabulary continuous speech recognition (CSR) by machine a very difficult one. Fortunately, speech is also structured and redundant: information about the linguistic content in the speech signal is often present at the various linguistic levels. To achieve acceptable performance, the recognition system must be able to exploit the redundancy inherent in the speech signal by bringing multiple sources of knowledge to bear. In general, these can include: acoustic-phonetic, phonological, lexical, syntactic, semantic and pragmatic knowledge sources (KS). In addition to designing representations for these KSs, methodologies must be developed for interfacing them and combining them into a uniform structure. An effective and coherent search strategy can then be applied based on global decision criteria. Practical issues that need to be resolved include computation and memory requirements, and how they could be traded off to obtain the desired combination of speed and performance.

In BYBLOS, we have explored many issues that arise in designing a large and complex system for continuous speech recognition. This paper is organized as follows. Section 2 gives an overview of the BYBLOS system. Section 3 describes our signal processing frontend. Section 4 describes the trainer system used for phonetic model knowledge acquisition. Section 5 describes the word model generator module that compiles word HMMs for each lexical item. Section 6 describes the syntactic/grammatical knowledge source that operates on a set of context-free rules describing the task domain to produce an equivalent finite state automaton used in the recognizer. Section 7 describes the BYBLOS recognition decoder using combined multiple sources of knowledge. Finally, Section 8 presents some figures and discussions on BYBLOS recognition performance.

2. Byblos System Overview

Figure 1 is a block diagram of the BYBLOS continuous speech recognition system. We show the different modules and knowledge sources that comprise the complete system, the arrows indicating the flow of module/KS interactions. The modules are represented by rectangular boxes. They are, starting from the top: Trainer, Word Model Generator, and Decoder. Also shown are the knowledge sources, which are represented by the ellipses. They include: Acoustic-Phonetic, Lexical, and Grammatic knowledge sources. We will describe briefly the various modules and how they interact with the various KSs.

Acoustic-Phonetic KS

The Trainer module is used for the acquisition of the acoustic-phonetic knowledge source. It takes as input a dictionary and training speech and text, and produces a database of context-dependent HMMs of phonemes.

Lexical KS

The Word Model Generator module takes as input the phonetic models database, and compiles word models phonetic models. It uses the dictionary - the lexical KS, in which phonological rules of English are used to represent each lexical item in terms of their most likely phonetic spellings. The lexical KS imposes phonotactic contraints by allowing only legal sequences of phonemes to be hypothesized in the recognizer, reducing the search space and improves performance. The output of the Word Model Generator is a database of word models used in the recognizer.

Grammatical KS

More recently, we have been working on representation and integration of higher levels of knowledge sources into BYBLOS, including both syntactic and semantic KSs. By incorporating both of these KSs into BYBLOS in the form of a grammar into our recognizer, we demonstrate improved recognition performance. In Section 6, we describe the Grammatical KS in more detail.

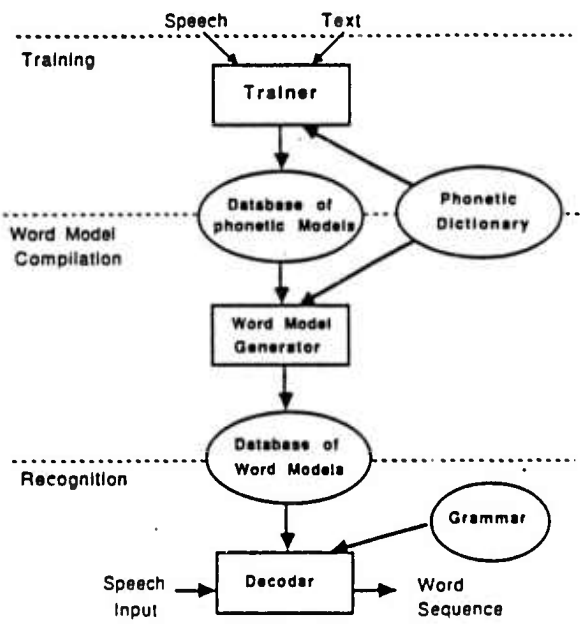


Figure 1: BYBLOS System Diagram.

3. Signal Processing and Analysis Component

The BYBLOS signal processing frontend performs feature extraction for the acoustic models used in recognition. Sentences are read directly into a close talking microphone in a natural but deliberate style in a normal office environment. The input speech is lowpass filtered at 10 kHz and sampled at 20 kHz. Fourteen Mel-frequency cepstral coefficients (MFCC) are computed from short-term spectra every 10 ms using a 20 ms analysis window. This MFCC feature vector is then vector quantized to an 8-bit (256 bins) representation. The vector quantization (VQ) codebook is computed using the k-means clustering algorithm with about 5 minutes of speech. We perform a variable-frame-rate (VFR) compression in which strings of up to 3 identical vector codes are compressed to a single observation code. We found this VFR procedure speeds up computation with no loss in performance.

4. Training/Acquisition Of Phonetic Coarticulation Models

The training system in BYBLOS acquires and estimates the phonetic coarticulation models used in recognition. Given that we model speech parameters as probabilistic functions of a hidden Markov chain, we make use of the Baum-Welch (also known as the Forward-Backward) algorithm [3] to estimate the parameters of the HMMs automatically from spoken speech and corresponding text transcription. For each training utterance, the training system takes speech and text, and builds a network of phonemes using the dictionary. It first builds the phonetic network for the word by using the phonetic transcription provided by the dictionary. The phonetic network is expanded into a triphone network so that each arc completely defines a phonetic context up to the triphone. These triphone networks of the word are then concatenated to form a single network for the sentence, which in general can take into account within word as well as across-word phonological effects. The training system then compiles a set of phonetic context models for each triphone arc in the network. It then runs the forward-backward algorithm to estimate the parameters of the phonetic context models. The Trainer operates in two modes: speaker-dependent and speaker-adapted. Associated with these two modes are two distinct methods for training the parameters of the hidden Markov models described below.

Speaker-Dependent

This is the algorithm used to find the parameters of the HMMs that maximizes the probability of the observed data given the model. This method produces HMMs that are finely tuned to a particular speaker, therefore in general would work well only for this speaker. Typically about 15 minutes of speech from a speaker is required for speaker-dependent training.

Speaker-Adapted

This is a new method of training that transforms HMM models of one speaker to model the speech of a second speaker [4]. This procedure estimates a probabilistic spectral mapping from a well-trained prototype speaker to a new speaker. Using this method it is possible for a new speaker to used the system with as little as 15 seconds of speech.

5. Word Model Generator

Prior to recognition, word HMMs are computed for each word in the vocabulary. The word model generator takes as input two objects: a database of phonetic HMMs as obtained in training, and a dictionary that contains phonetic spellings for each word. For each phoneme in each word of the lexicon, it first finds in the phonetic HMM database all the context models that are relevant to this phoneme in its particular phonetic environment. It then combines this set of phonetic models with appropriate weights to produce a single HMM for each phoneme in the word. This combination process saves computation by precompiling the many levels of phonetic context models that can occur for a given phonetic context into a single representation. The output of the word model generator is a database of word HMMs serving as the input to the decoder.

6. Grammatical Knowledge Source

To solve the CSR problem requires major advances in two areas: acoustic modeling and language modeling. A good acoustic model is essential in making fine phonetic distinctions when needed. However, it is not sufficient by itself to solve the CSR problem. In a complex task with large vocabulary where the number of hypothesized word candidates is large, the probability for acoustic confusability can be high, and the recognizer could make errors. A conceptually simple yet effective way to restrict the number of words that are allowed to be hypothesized, and therefore decrease probability of acoustic similarity, is to incorporate a grammar into the recognizer. It is well known that recognition performance improves as vocabulary size decreases. Similarly, when syntactical information is used to reduce the number of words that can legally follow a given sequence of words, a recognizer is expected to make fewer errors. The purpose for using a grammar then, is to improve recognition performance, with an added benefit of reducedcomputation.

Grammar Design and Implementation

We approach the implementation of a grammar in BYBLOS in two stages. First, we create a description of the task domain language using a modified context-free notation. Typically this description is based on a representative set of sentences that characterizes the task domain, and is designed to capture generalizations of the linguistic phenomena found in them. Second, we use a tool that transforms this description into structures in our recognizer that provide the corresponding grammatical constraints. This tool provides us with a general facility for capturing in BYBLOS an approximation of any language expressible in context-free grammars (CFG) expressed as context-free rules. We elected to implement the grammatical constraints in the form of a finite state automaton (FA) similar to those described in [5].

At the first stage in generating a grammar, we use a context-free notation augumented with variables in order to simplify the process of describing a language. For example, this notation would allow a rule that says a noun phrase of any number can be replaced by an article and a noun of the same number; ordinary context-free notation would require two rules that are identical except that one would be for singular number and the other for plural.

Our system first translates the augmented notation into ordinary CFGs and then constructs a FA based on these rules. Because context-free grammars can accept recursive languages and a FA cannot, recursion is approximated in the FA by limiting the number of levels of recursion. Such an approximation is reasonable for most task languages, since spoken sentences do not ordinarily use more than a few levels of recursion.

7. Recognition Search Strategy

Once the FA is compiled from the context-free description of the task domain, it is ready to be used in the decoder. An important characteristic of a recognizer is the search strategy that is used to find the word sequence that best matches the input speech. We believe that an optimum search strategy avoids making local decisions; the search decision should be made globally, based on scores from all the KSs. One such search paradigm is the one used in BYBLOS: the search is made top down, linguistically driven, with tightly coupled KSs.

The FA is convenient for deploying such a search strategy. It is used as follows in our recognizer. We associate with each transition in the FA a hidden Markov model for the word. This model is used to compute the probability of the acoustic event (sequence of VQ spectra) given the occurrence of the word at that place in the grammar. Before the start of recognition, the initial state of the FA where a legal sequence of words can begin is initialized to unity, and all the other states are initialized to zero. For each 10 ms frame of the input speech, the scores for the states in all the words in the FA network are updated using modified Baum-Welch algorithm [2]. In addition to state updates within a word, a word can have a score propagated to its initial state from its best scoring predecessor word. This simple state update operation is repeated every 10 ms for each FA transition until the end of the utterance is reached. The decoder output is then computed by tracing back through the FA network to find the highest scoring sequence of words that end in the terminal state of the FA.

One potential problem associated with using a FA grammar for recognition is that computation is expected to be proportional to the number of transitions in the FA. This number can be quite large for complex languages. However, in our experience with different grammars in our recognizer, we find that a beam search effectively reduces the computation to a very manageable level while maintaining the same performance as that of an exhaustive search.

8. Byblos Recognition Performance

In [2], we presented word recognition results for a 334-word electronic mail task. In speaker-dependent mode, we demonstrated performance of 90% across several speakers without the use of a grammar (i.e., branching factor of 334). Since then, we have tested the system along many dimensions: two task domains, FA grammars with varying perplexities, varying amounts of adaptation speech, and different speaker types. The results are tabulated in Figure 2. Below we describe the different conditions in more detail.

Task Domains

The two task domains tested are: Electronic Mail (EMAIL) and Naval Database Retrieval (FCCBMP). Both tasks have vocabulary sizes of approximately 350 word (334 for EMAIL, 354 for FCCBMP). A description of the task domain language was created using CFG. The CFGs were designed to capture generalizations of linguistic phenomena found in example task domain sentences.

Grammars

Two finite state grammars were generated for each task domain: Command and Sentence. The Command Grammar in each case was designed to cover only the command subset of

Grammar/ Perplexity / Training Time	EMAIL		FCCBMP	
	Command (20)	Sentence (30)	Command (22)	Sentence (30)
15 minute	98.4	98.8	99.6	99.5
2 minute	97.9	94.9	96.6	96.2

Figure 2: BYBLOS Recognition Results. Two task domains (EMAIL and FCCBMP), two grammars for each task (Command and Sentence), and varying amounts of training speech (2 minutes and 15 minutes). Also shown are maximum perplexity measures for the grammars.

the language; the Sentence Grammar was designed to cover all of the language, which included both command and question type constructs. The maximum perplexity measures of the grammars, as proposed in [6], are shown in Figure 2. In both tasks, the sentence grammars have a higher perplexity than their command counterparts.

Adaptation Time

As described in Section 2, The BYBLOS operate in two modes, speaker-dependent and speaker-adapted. In speaker-dependent mode, 15 minutes of training speech is required for a speaker. This mode in general will give word accuracy in the 98.5+ range. In the speaker-adaptive mode, anywhere from 2 minutes down to 15 seconds of speech from a new speaker is needed to "adapt" the HMM parameters to the new speaker. The performance in this case is 97%.

Speaker Type

We have tested BYBLOS on several speakers with different dialects, including a female speaker, a non-native speaker, and 3 naive (uncoached) speakers. The recognition results for these speakers showed little deviation typical male speakers of standard American dialects.

9. Summary

We have presented BYBLOS, a system for large vocabulary continuous speech recognition. We showed how we integrate multiple sources of knowledge to achieve high recognition performance. In recognition experiments, we demonstrated consistent performances across task domains, grammars, adaptation time, and speaker type.

We are currently working to improve various aspects of the system, including: a real time implementation of the recognizer, search strategy, acoustic modeling, and language modeling. In the future, we plan to work on integration of speech and natural language for speech understanding applications.

Acknowledgement

This work was supported by the Defense Advanced Research Projects Agency and was monitored by the Space and Naval Warfare Systems Command under Contract No. N00039-85-C-0423.

References

1. R.M. Schwartz, Y.L. Chow, O.A. Kimball, S. Roucos, M. Krasner, and J. Makhoul, "Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Tampa, FL, March 1985, pp. 1205-1208, Paper No. 31.3.

2. Y.L. Chow, R.M. Schwartz, S. Roucos, O.A. Kimball, P.J. Price, G.F. Kubala, M.O. Dunham, M.A. Krasner, and J. Makhoul, "The Role of Word-Dependent Coarticulatory Effects in a Phoneme-Based Speech Recognition System", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Tokyo, Japan, April 1986, pp. 1593-1596, Paper No. 30.9.1.

3. L.R. Bahl, F. Jelinek, and R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 2, March 1983, pp. 179-190.

4. R.M. Schwartz, Y.L. Chow, G.F. Kubala, "Rapid Speaker Adaptation using a Probabilistic Spectral Mapping", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Dallas, TX, April 1987, Elsewhere in these proceedings

5. R.G. Goodman, *Analysis of Languages for Man-Machine Communication*, PhD dissertation, Carnegie-Mellon University, May 1976.

6. M.M. Sondhi and S.E. Levinson, "Computing Relative Redundancy to Measure Grammatical Constraint in Speech Recognition Tasks", *IEEE Int Conf Acoust., Speech, Signal Processing*, Tulsa, OK, April 1978, pp. 409-412.

Presented at the IEEE International Conference on Acoustics, Speech, and Signal Processing, Dallas, Texas, April 6-9, 1987.

Efficient Implementation of Continuous Speech Recognition on a Large Scale Parallel Processor

Owen Kimball, Lynn Cosell,
Richard Schwartz, Michael Krasne.

BBN Laboratories
10 Moulton St.
Cambridge, MA 02238

Abstract

This paper presents research into the use of large-scale parallelism for a continuous speech recognition algorithm. The algorithm, developed for the BBN Byblos system [1], uses context dependent Hidden-Markov models to achieve high recognition accuracy. The multiprocessor used in the research, the BBN Butterfly™ Parallel Processor, is a shared memory, MIMD machine. The algorithm was implemented using the Uniform System software methodology, a system that simplifies parallel programming without sacrificing efficiency. The algorithm is described, highlighting those portions critical to an efficient parallel implementation. Some of the problems encountered in trying to improve efficiency are presented as well as the solutions to those problems. The algorithm is shown to achieve 79% processor utilization on a 97-node Butterfly Parallel Processor. This is equivalent to a speedup by a factor of 77 over a single processor benchmark.[1]

1. Introduction

The introduction of large-scale parallelism in computers offers the potential for greatly increased speed and better performance-cost ratios for algorithms that can make use of this parallelism. This paper describes the parallel implementation of a continuous-speech recognition algorithm that successfully uses the speedup provided by a general purpose multiprocessor, the Butterfly Parallel Processor.

The outline of this paper is as follows: Section 2 describes the Butterfly Parallel Processor and the Uniform System, Section 3 describes the BBN word recognition algorithm, Section 4 explains the initial parallel implementation of the algorithm, Section 5 describes the improvements to the algorithm for better processor utilization and presents results based on these improvements. The final section presents some conclusions from the work.

2. Butterfly and Uniform System

The Butterfly Parallel Processor [2] is composed of multiple (up to 256) identical nodes, each containing a processor and memory, interconnected by a high-performance switch. The Butterfly architecture is multiple-instruction-multiple-data-stream (MIMD), in which each processor node executes its own sequence of instructions, referencing data as specified by the instructions. Each processor node contains either a Motorola MC68000 or MC68020 microprocessor, an optional floating-point co-processor, from 1 to 4 megabytes of main memory, a co-processor called the Processor Node Controller, memory management hardware, an I-O bus, and an interface to the Butterfly switch.

The Butterfly switch allows each processor to access the memory on every other node. Collectively, these memories form the shared memory of the machine, a single address space accessible to every processor. All interprocessor communication is performed using shared memory. Instructions accessing memory on the same node as a processor typically take about 2 microseconds to complete, whereas those accessing memory on another node take about 5 or 6 microseconds. Block transfers from one memory to another run at 4 megabytes per second. The machines used in this project were 16-processor and 97-processor machines, each with 1 megabyte of memory and a MC68000 microprocessor on the processor nodes. Neither had hardware support for floating point arithmetic.

The software for the project was written using the Uniform System, a programming methodology supported by a library of high-level subroutines [3]. The benefit of using the Uniform System is that it can provide a simple, efficient solution to the problem of load balancing for the memory as well as for the processors. To balance the load on memory, the Uniform System routines spread out the data evenly across the different physical memories in the machine. Under the assumption that distributed data will also distribute memory accesses fairly evenly, this approach can reduce the inefficiency that results when many processors attempt to access the same memory simultaneously.

To balance the load on processors, the Uniform System treats processors as a pool of identical workers, all of which can execute the same tasks. In this way, tasks can be dynamically assigned to the free processors in the machine. In a typical program, control starts out in a single processor of the machine. To perform tasks in parallel, this processor calls a Uniform System "generator" subroutine, specifying a set of tasks to do and a task subroutine. The generator creates a descriptor of the work to be done and starts all processors. The processors then perform the work in parallel, each taking the next task data from the descriptor and executing the task routine with this data until all the work is completed. At that point, control is returned to the original single processor. An

[1]This work was sponsored by the Defense Advanced Research Projects Agency and was monitored by the Space and Naval Warfare Systems Command under Contract No. N00039-85-C-0313.

The authors would like to thank William Crowther for his assistance on the binary-tree maximum and for other informative discussions.

example of a simple generator is GenOnI. The call "GenOnI(task_routine, Ntasks)" assigns processors to perform the subroutine "task_routine" for every integer value in the range 1 to Ntasks.

3. Recognition Algorithm

The Byblos system has two major components, a trainer and a recognizer. The recognition component was implemented on the Butterfly Parallel Processor. The training component uses the forward-backward algorithm [4] to estimate discrete-density Hidden-Markov models of context-dependent phonemes. It combines these models to form word models that are used in recognition. The context-dependent models lead to accurate and robust recognition performance; the system has achieved 90% correct recognition on a 335 word speaker-dependent task with no grammar [5].

In the recognition process, input speech is analyzed every 10 ms and then vector quantized with a 256-vector codebook. The analysis and quantization are done in real time on an FPS array processor attached to a VAX. The quantization codes, each representing a frame of input speech, are input in real time over an ethernet connection to the search algorithm on the Butterfly Parallel Processor

The search algorithm finds the best scoring sequence of words using the trained word models. Each possible sequence of words that is considered is called a word sequence theory. The search uses the Viterbi decoding algorithm to update scores for all word sequence theories at each frame. In order to prevent underflow during score updating, all theory scores are normalized. To determine the "normalization factor" for a frame, the algorithm computes the maximum score of all states in all words in the frame and sets the factor to the score ceiling minus the maximum score.

The major work being performed in the algorithm can be abstracted in pseudo-code as follows:

```
FOR all input frames {
   max_score := 0
   best_end_score := 0
   FOR all words {
      update word score
      IF word_max_score > max_score
         max_score := word_max_score
      IF word_end_score > best_end_score
         best_end_score := word_end_score
   }
   determine initial state score for new theories from best_end_score
   determine normalizaton from max_score
}
determine and report best scoring theory
```

The algorithm computes two maxima: "max_score", the maximum over all states of the words scored in an input frame, and "best_end_score", the maximum score of all words' *final* states. The first maximum is used for the normalization factor mentioned above, and the second is used to determine the score for the initial state of all words in the next frame.

The core of this computation, the word score update, entails updating all the phonemes in a word. Each phoneme update requires a little less than one millisecond of computation, and the average word update time is slightly more than 4 milliseconds for the vocabularies used in this work.

4. Initial Parallel Implementation

As the first step toward a parallel implementation, the speech recognition program was ported from VAX/VMS to a single processor of the Butterfly Parallel Processor. Both versions of the program were in the language 'C'. The most significant change to the program in this phase was the use of the Uniform System memory management routines to store in global shared memory about 1.5 megabytes of data that had been stored on disk in the VAX version.

The VAX (and the first Butterfly System implementation) used floating-point arithmetic, but because floating-point arithmetic is performed in software in our Butterfly Parallel Processor, it is substantially slower than fixed point. For this reason, the program was switched to fixed-point arithmetic. As part of this change, multiplication of probabilities in the original version was converted to addition of corresponding log probabilities. With this modification, the execution time was about two minutes for a 3.5 second utterance from a 120 word task. This is about the same speed as our optimized floating point VAX 11/780 program.

Examination of the pseudo-code in the preceding section leads to a natural decomposition of the algorithm: the fundamental parallel task is to update the score of a single word for a single input frame. Using the Uniform system generator GenOnI, the pseudo-code for the parallel version of our algorithm becomes:

```
FOR all frames {
   best_end_score := 0
   max_score := 0
   GenOnI(update_word, N_words)
   determine initial state score for new theories from best_end_score
   determine normalizaton from max_score
}
determine and report best scoring theory
```

In this version, the subroutine update_word now includes the calculation of max_score and best_end_score. Note that since the processor calling GenOnI waits for all processors to finish before proceeding, this mechanism provides a synchronization that is needed to ensure that no processor begins updating words of a new frame until the initial state score and the normalization factor for the next frame have been computed.

Using this simple approach to parallel implementation, a first timing experiment was conducted using a 16-processor machine and a 120-word vocabulary task. Processor utilization was found to be 75%, i.e. the machine was effectively using the computation corresponding to 12 of the 16 actual processors. [6]. This result was judged to be good enough to proceed directly to work on a larger machine. The first time the program was run on a 97-processor machine, processor utilization was approximately 20%. Although this represents a factor of 20 speedup of the program, it is an inefficient use of the machine. The next section presents several factors that contributed to the inefficiency as well as the methods used to improve them.

5. Efficiency Improvements and Results

There are a number of potential obstacles to attaining efficient processor utilization on a multiprocessor. Typical issues include contention for a common memory location, serial code in the program, and processors waiting idly to synchronize with other processors. Each of the specific problems described below includes one or more of these issues.

Number of Tasks and Startup Overhead

Even before the program was run on a larger machine, we had anticipated that it would be hard to obtain high processor utilization with a vocabulary as small as 120 words. Since our long-term goal is to recognize speech from large vocabularies, we switched to a larger task of 335 words. This change improved processor utilization to 35% on the 97-processor machine.

The speed of processor scheduling was examined next. In the initial parallel version shown above, the generator subroutine call starts all the processors at each frame. It was found that the overhead of starting up was relatively large for the amount of work being done at each frame. To reduce the overhead, the program was altered to start all processors only once at utterance start, generating *Nframes*×*NWords* tasks at that point and letting each processor determine its word and frame indices from the single task index it receives from the generator. Processor utilization improved to about 50% with this change.

Processor Synchronization Issues

The task generation change had removed the synchronization provided by starting up a new generator at each frame. To replace this, an explicit synchronization was built into the program to be performed after all the words in a frame were processed. There were two subsequent changes to improve the efficiency of synchronization. The first dealt with task ordering. In the early versions of the algorithm, processors updated all the words in the vocabulary, with no particular ordering of the words. Since words have varying numbers of phonemes (from one to 14 phonemes in this task's vocabulary), different words took different amounts of time to update. If a processor began work on a long word near the end of the work for a frame, other processors would finish their assigned words and wait idly to synchronize with the one busy processor. To reduce this inefficiency, the words were processed in order from longest to shortest (in number of phonemes).

In figure 1, we schematically depict the situation before and after the words are ordered. The filled rectangles represent time when processors actively work on tasks and the white space represents time between tasks when no work is being accomplished. In the right hand part of the figure, idle processor time is substantially reduced by sorting.

The second change to synchronization efficiency concerned the point in the program at which synchronization was done. As mentioned, the purpose of the synchronization was to ensure that no processor proceeded to the next frame until the starting score for words and the normalization factor were computed. Since the normalization factor was only to avoid score underflow, it could be estimated a frame or more earlier. The only remaining synchronization constraint was the word-starting score. This score, however, is used only at the beginning of the *first* phoneme of each word. Considering this, the order of the update of a word was reversed so that the last phoneme was updated first, and the first phoneme updated last.

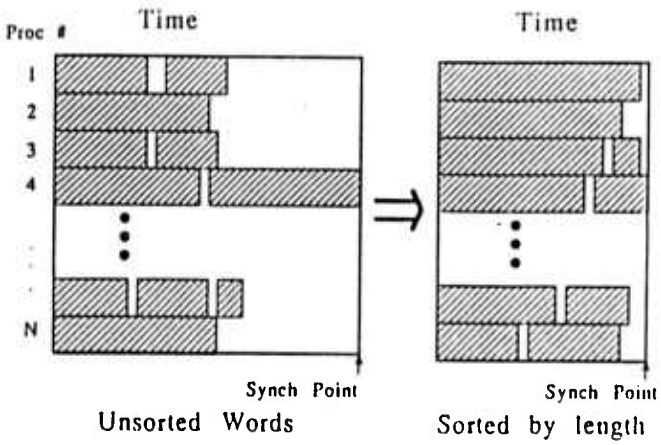


Figure 1: Ordering Tasks by Length

This change allowed a processor to finish work on one frame and immediately begin work on updating a word from the next frame, synchronizing only when it got to the first phoneme. In this way, time that had been previously spent by processors waiting for others to finish a frame was now being used to perform useful work from the next frame.

Figure 2 depicts the situation for two frames of an utterance before and after this change. Tasks for time T+1 are shown in two shades. The darker portion represents the part of the task that depends on the previous frame's work being finished. On the right, with the the order of the computation reversed, the idle processor time is reduced. The effect of the synchronization changes was to increase processor utilization to approximately 72%.

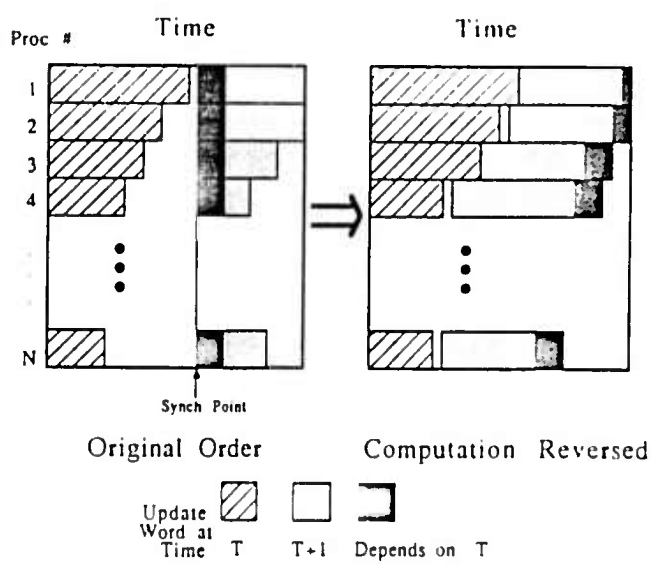


Figure 2: Reversing Word Computation Order

Finding Global Maximum

Finally, the efficiency of finding the maximum value was also improved. A straightforward computation of the maximum value requires that all values be compared with a single memory location, but this approach results in contention for that location. As a first improvement, the program was altered to make each processor maintain its own *local* maximum of the scores of all the words that it updates in a frame. At the end of the frame, the global maximum of these values over all processors was determined. In initial versions, this was accomplished by having processors sequentially compare their value to the global location and replace it if necessary. Although on a sixteen processor machine, the time for processors to turn in values in this way is negligible, with 97 processors, the inefficiency of the approach becomes noticeable.

An alternative to this approach was to set up a "binary tree" of locations for taking the maximum. In this approach, the processors' local maxima are the leaves of the tree and the maxima are propagated up through the nodes of the tree. This approach reduces the asymptotic time for finding a global maximum from O(N) to O(log N), where N is the number of processors. More importantly in our case, efficiency improved because memory contention was reduced.

The total effect of all the improvements described above was to improve processor utilization on a 97-processor machine from 20% to 79%. Figure 3 is a graph of processor utilization for 1 to 97 processors on the 335 word task. The actual speed of the speech recognition improved accordingly. After the optimizations are included, a one-processor Butterfly Parallel Processor requires 128 times real time (128 seconds to process one second of input speech) and a 97-processor machine requires about 1.7 times real time.

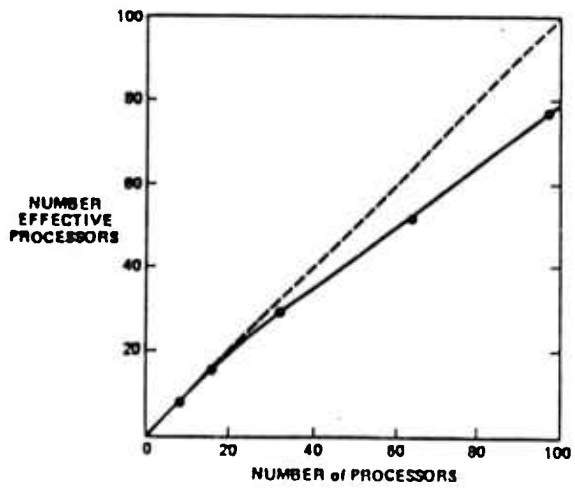


Figure 3: Butterfly Processor Utilization, 335 words

6. Conclusions

This work has shown that the Butterfly architecture is suitable for continuous speech word recognition. The algorithm was implemented efficiently without changing the type or amount of computation performed. Some ingenuity was required to obtain an efficient realization, but once the obstacles were understood, solutions presented themselves fairly readily. The memory and processor management functions of the Uniform System made initial parallelization of the algorithm quite easy and provided several alternatives for improving implementation efficiency when required.

We draw several broad conclusions about efficient parallel programming as well. Most obviously, and perhaps most importantly, it is crucial that sequentially executed code be eliminated wherever possible. Similarly, much of the inefficiency in our original multiprocessor program was due to processors waiting for each other. Synchronizing processors only after all other possible work is done was found to be a good strategy to avoid this. Additionally, it can be very important to minimize the overhead of parallel constructs such as starting processors.

References

1. Y.L. Chow, M.O. Dunham, O.A. Kimball, M.A. Krasner, G.F. Kubala, J. Makhoul, P.J. Price, S. Roucos, and R.M. Schwartz, "BYBLOS: The BBN Continuous Speech Recognition System", *ICASSP*, Dallas, TX, April 1987, Elsewhere in these proceedings

2. R. Thomas, R. Gurwitz, J. Goodhue, and D. Allen, "Butterfly Parallel Processor Overview", Tech. report, BBN, March 1986.

3. R. Thomas, "The Uniform System Approach to Programming the Butterfly Parallel Processor", Report 6149, BBN, June 1986.

4. L.E. Baum and J.A. Eagon, "An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model of Ecology", *Amer. Math Soc. Bulletin*, Vol. 73, 1967, pp. 360-362.

5. Y.L. Chow, R.M. Schwartz, S. Roucos, O.A. Kimball, P.J. Price, G.F. Kubala, M.O. Dunham, M.A. Krasner, and J. Makhoul, "The Role of Word-Dependent Coarticulatory Effects in a Phoneme-Based Speech Recognition System", *ICASSP*, Tokyo, Japan, April 1986, pp. 1593-1596, Paper No. 30.9.1.

6. L.K. Cosell, O.A. Kimball, R.M. Schwartz, M.A. Krasner, "Continuous Speech Recognition on a Butterfly Parallel Processor", *Proceedings of the International Conference on Parallel Processing*, St. Charles, Illinois, August 1986, pp. 717-720.

A NEW MODEL FOR THE TRANSDUCTION STAGE OF THE AUDITORY PERIPHERY*

Stephanie Seneff

Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

A new model is proposed for the transformation in the cochlea from Basilar membrane vibration to nerve fiber responses in the VIIIth nerve. The model has been incorporated into a system for speech processing that we are currently using as a front end in a speech recognition system under development. We have found that spectral representations based on this model show certain advantages over traditional methods for spectral analysis for particular applications.

We believe that this model represents the auditory periphery much more accurately than previous models that we have used. The most important change is that a new adaptation model is used, one that was originally proposed by Goldhor [1]. With this adaptation model it is possible to obtain reasonable matches to the equal-incremental-response criterion imposed by the Smith and Zwislocki data [2]. Parameters of the system were adjusted so as to match this criterion as well as possible. In addition, the model was compared with auditory data in four other experimental categories as well, as discussed in the paper. These categories were selected because we believe they reflect important aspects of the response for speech applications. Examples are given of outputs of the system for speech signals in order to illustrate how the nonlinearities in the model affect the responses to speech.

INTRODUCTION

The peripheral auditory system is typically modelled by a bank of linear filters which resemble available data on the shapes of auditory filters, followed by a nonlinear stage that attempts to capture the dynamics of the transformation from Basailar membrane vibration to nerve fiber response. This part of the model incorporates such nonlinearities as dynamic range compression and half-wave rectification, and also captures effects such as short-term adaptation, rapid adaptation, and forward masking. It is very difficult to devise a scheme that will accurately reproduce diverse aspects of auditory response, yet we feel that it is very important in speech processing for these aspects to be more-or-less correct. An enormous amount of data is available from measurements made from auditory nerve fibers for a number of different experimental paradigms. A useful goal is to attempt to reproduce gross features that emerge from such experiments. We selected auditory data from five different categories of response measurements to be compared with the model. These demonstrate the degree of success in capturing the detailed wave shape in steady state conditions, the dynamics of onset response, the degree of forward masking, the extent of loss of synchrony at high frequencies, and the incremental response characteristics at onset and steady state.

MODEL DESCRIPTION

Figure 1 shows a block diagram of our current auditory-based front-end system. The initial stage is a bank of linear filters, which is followed by the "hair-cell/synapse" stage that introduces the nonlinearities. A bifurcation of the outputs leads to two spectral-like representations, the "Envelope Spectrogram" and the "Synchrony Spectrogram." The only part of this model that has been changed since previous reports [3] is the hair-cell/synapse stage. The new model for this stage consists of four subcomponents, as shown in Figure 2: a half-wave rectifier, a short-term adaptation circuit, a lowpass filter, and a rapid Automatic Gain Control (AGC). We will discuss each of these components in turn.

All of these components except the lowpass filter are nonlinear, and therefore the final output is affected by the *ordering* of the components. A particular ordering can be justified in part by forming associations with elements of

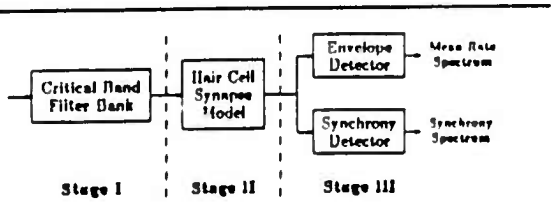


Figure 1: Block diagram of our computer model

*This research was supported by DARPA under Contract N00039-85-C-0254, monitored through Naval Electronic Systems Command.

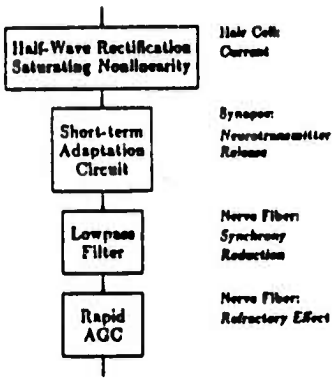


Figure 2: Block diagram of the subcomponents of Stage II with suggested auditory system affiliations indicated at right

the actual auditory system. Such links can also aid in the design of each individual component. To the right of each component in the figure is proposed a corresponding affiliation with the auditory system. The hair-cell current response as measured for amphibians shows a distinct directional sensitivity [7]. It is not clear that the current is a direct link in the response mechanism; nonetheless, it is tempting to assume that half-wave rectification first occurs in the hair cell, and hence this is the first component in the model. There seems to be no evidence for short-term adaptation in hair cell current or voltage responses; therefore it is generally assumed that this effect is introduced in the synapse between the hair cell and the neuron [5]. The logical ordering is therefore to place this component second.

The AGC is assumed to be affiliated with the refractory phenomenon of nerve fibers; therefore, this component should be placed late in the series. Such an affiliation implies that the rapid adaptation component of responses to onsets is due to the refractory phenomenon, a theory that has been proposed by Johnson and Swami [6]. It is difficult to know where to place the lowpass filter. It is associated with the gradual loss of synchrony in nerve fiber responses as stimulus frequency is increased. The locus (or loci) of such synchrony loss has not yet been determined. The lowpass filter must follow the half-wave rectifier, because it only makes sense after signal energy has been preserved through a DC component. The solution adopted was to try placing the lowpass filter in all three of the remaining positions, and choose the one that yields the best behavior in the final response.

The model for the half-wave rectifier, whose response function is shown in Figure 3, is defined mathematically as follows:

$$\begin{array}{ll} 1 + A \tan^{-1} Bx & x > 0 \\ e^{ABx} & x \leq 0 \end{array} \qquad (1)$$

It is thus exponential for negative signals, linear but shifted

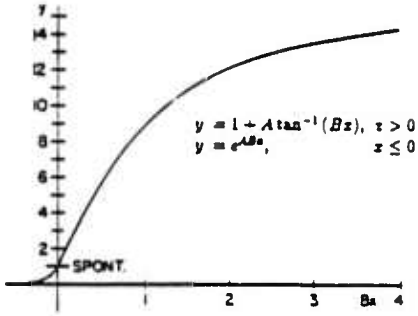


Figure 3: Plot of input-output response of the half-wave rectifier used in the model. This mapping resembles the hair cell current response as measured for frogs [4]

(by a "spontaneous" rate of 1) for small positive signals, and compressive for larger signals, saturating at $1 + A\pi/2$. It is based on the measured hair cell current responses as a function of a fixed displacement of the cilia as determined in frogs by Hudspeth and Corey [7].

The model for short-term adaptation is very similar to one proposed by Goldhor [1]. It consists of a simple nonlinear circuit, as shown in Figure 4. The input is the voltage source, V_i, and the output is the current through the conductance $G1$. $G1$ is in series with a diode, which serves to lock out this branch of the circuit whenever the voltage across $G1$ becomes negative (the "off" condition). There is another conductor, $G2$, in parallel, in addition to a capacitor. The capacitor accumulates a charge whenever the signal V_i is sufficiently positive, and discharges through G_2 when the stimulus voltage falls below the capacitor voltage.

Goldhor showed that such a circuit, when applied using the *envelope* of the stimulus as the input V_i, obeys the equal incremental response property of short-term adaptation [2], and also appropriately exhibits a longer time constant for recovery after signal offset than for adaptation after signal onset. The latter property holds because

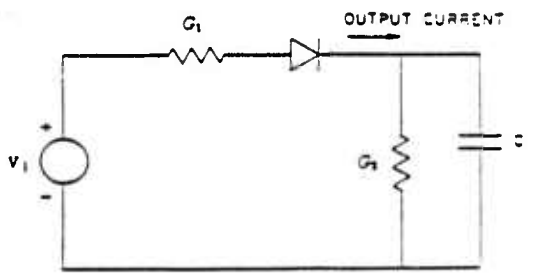


Figure 4: Goldhor's [1] adaptation circuit. In our model, the output of the half-wave rectifier of Figure 3 is used as V_i, the input to the adaptation circuit.

during recovery the diode is turned off, and the capacitor discharges only through G_2, whereas after increments the diode tends to be on, and current can flow through both conductors to charge the capacitor more quickly.

Our model uses the same circuit, except that the detailed cycle-by-cycle behavior of the input signal is preserved in V_i. The consequence is that the diode turns on and off for each period of the stimulus, and an adapted response is obtained only after the capacitor reaches a steady state condition in which the amount of charge gained during the time in which the input voltage level is higher is exactly the same as the amount lost during the remaining portion of the cycle. One consequence is that the effective time constant for adaptation lies somewhere between the "on" time constant, τ_1, and the "off" time constant, τ_2. The time constant for recovery, on the other hand, is equal to τ_2.

The current through the diode branch of the adaptation circuit is next processed through a lowpass filter that achieves two important effects: it reduces synchrony to high-frequency stimuli and it smooths the square wave shape encountered in the half-wave response for saturating stimuli. The lowpass filter was realized as a cascade of n_{LP} leaky integrators, each with an identical time constant τ_{LP}. The two parameters, n_{LP} and τ_{LP} were adjusted to match available data on synchrony loss [8].

The final component is the rapid AGC, which is defined as follows:

$$y[n] = \frac{x[n]}{1 + K_{AGC} < x[n] >} \qquad (2)$$

where K_{AGC} is a constant and <> symbolizes "expected value of," obtained by processing $x[n]$ through a first-order lowpass filter, with time constant τ_{AGC}. This equation resembles in form the formula obtained theoretically by Johnson and Swami [6] as a steady-state solution for a simple model of the refractory effect, where it is assumed that a response is locked out for a time interval Δ after a spike occurs:

$$y(t) = \frac{x(t)}{1 + \int_{t-\Delta}^{\Delta} x(\alpha)d\alpha} \qquad (3)$$

Figure 5 shows the outputs of intermediate stages of the 2000-Hz channel in response to a high-amplitude tone at CF. The envelope of the response over a long time interval is shown on the left, and the detailed wave shapes near tone onset are shown on the right. Part *a* shows the response after only the linear filter of Stage I. Part *b* shows the response after the instantaneous half-wave rectifier. The square wave shapes introduced here are lost after the lowpass filter. The effects of the short-term adaptation component are apparent in the envelope response on the left in part *c*. The final AGC further alters the dynamics

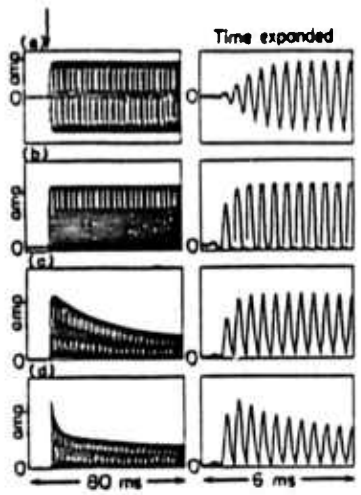


Figure 5: Responses at intermediate stages of the hair-cell/synapse model for 2000-Hz tone at CF at high signal level: (a) after critical band filter, (b) after half-wave, (c) after short-term adaptation and lowpass filter, and (d) after AGC

of the onset, to produce a trend quite typical of auditory nerve fibers, as shown in part *d*.

COMPARISONS TO AUDITORY DATA

The above system has a number of parameters that can be adjusted according to some criteria based on relevant auditory data from the literature. At the same time, the degree of success in the matching process can help to evaluate the model's adequacy for capturing auditory phenomena. The following data were selected as responses that should be matched, in part based on a judgment of which aspects of the auditory response are likely to be significant with regard to speech analysis.

- Envelope response characteristics at tone onsets as a function of tone level,
- Forward masking effects as a function of masker level
- Period histogram responses in steady state conditions for one-formant vowel stimuli, as a function of stimulus level,
- Equal incremental response characteristic, and 5:2 onset-to-steady-state ratio, and
- Synchrony falloff characteristics as a function of tone frequency.

The parameters of the system were adjusted to match all of the above criteria as well as possible. Several iterations through the matching process were necessary for

half-wave		adaptation		lowpass		AGC	
A	G_{HW}	τ_1	τ_2	τ_{AGC}	K_{AGC}	τ_{LP}	n_{LP}
10	2.35	15 ms	120 ms	3 ms	.002	.04 ms	4

Table 1: Fixed parameter values used for experiments

convergence. Some surprising results emerged from the exercise; most remarkable was that τ_2 for the Goldhor adaptation circuit had to be set to a much larger value than was anticipated in order to match the forward masking data. Another discovery was that, although the short-term adaptation component and the AGC component interact in a complex way, it is possible to set their paramaters so that the equal-increment criterion imposed by the Smith and Zwislocki experiment is reasonably well matched. We will discuss each of the above criteria in turn, in each case showing a plot of the auditory data and the corresponding model response. The output of the half-wave rectifier was multiplied by a gain term, G_{HW}, which was adjusted to yield a final output that could be equated with a firing rate. In all cases, the various time constants of the model were set at fixed values, according to Table 1.

Tone Onsets:

Delgutte [9] plotted the envelopes of responses of cat's ear nerve fibers to tone bursts as a function of eight different tone levels, as shown here in Figure 6a. The experimental paradigm was reproduced for the computer model, and the resulting responses are shown in Figure 6b. Onset response characteristics are largely dominated in the model by the parameters of the rapid AGC component.

Forward Masking:

Delgutte's [9] plots for a forward masking experiment are shown in Figure 7a, along with the results of the computer model in Figure 7b. The plots are given as a function of adapter level, with the test tone level held fixed. The main controlling factor of forward masking in the model is τ_2 of the short-term adaptation circuit.

Period Histograms:

Delgutte's [9] plots of the period histograms of responses to a one-formant vowel stimulus are shown in Figure 8a, along with the model results in Figure 8b. In both the auditory data and the model data, the formant bandwidth in the response appears to become larger (more rapid decay with each period) at intermediate amplitudes, and much smaller at large amplitudes, when saturation effects are dominating the response. The half-wave rectifier is the controlling factor in this steady-state phase-locked response characteristic, although the short-term adaptation circuit also plays a role.

Figure 6 Figure 7

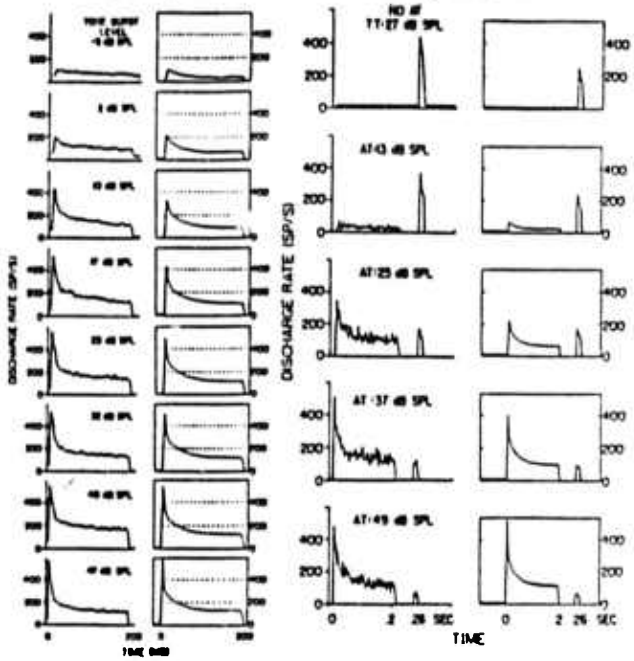


Figure 6: (*left*) Response patterns of an auditory nerve fiber to a tone burst as a function of signal level (from Delgutte [9]). The 180-ms burst has a rise/fall time of .25 ms, and a frequency, 770 Hz, approximately equal to the fiber CF. The post-stimulus-time (PST) histogram was computed with a bin width of 1.4 ms and then smoothed with a three-point smoother.

Figure 7: (*left*) Response patterns of an auditory nerve fiber to a 20-ms test tone preceded by a 200-ms adapting tone (from Delgutte [9]). Both tones have a rise time of 2.5 ms, and a frequency, 1220 Hz, approximately equal to the fiber CF. Histograms are computed with a 1-ms bin width, and three-point smoothed. (*right*) Response patterns for the computer model for the same stimulus conditions, using a 3-ms Hamming window for smoothing.

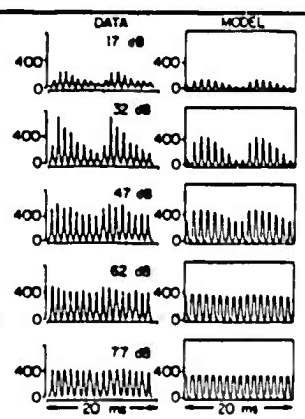


Figure 8: (*left*) Response patterns of an auditory nerve fiber to a single-formant synthethic stimulus as a function of signal level (from Delgutte [9]). The stimulus has an 800-Hz formant frequency, approximately equal to the fiber CF. Formant bandwidth is 70 Hz, and the fundamental frequency of voicing is 100 Hz. The 10-ms period histogram, computed with a 50-μs bin-width, is repeated twice in each case, to show two pitch periods of the response. (*right*) Response patterns for the model for the same stimulus conditions. The responses in this case are unsmoothed.

Incremental Responses:

Smith and Zwislocki [2], using tone pedestals as stimuli, measured rate responses of guinea pig auditory nerve fibers as a function of time. The stimuli consisted of sudden-onset tone bursts whose amplitudes, I, were incremented by an amount δI at a time $\tau = 150$ ms after initial onset. A PST histogram of the response was computed, and a difference between the response just before and just after the amplitude increment constituted a "steady state incremental response." This incremental response, defined by $IR = R_\tau^+ - R_\tau^-$, was then compared with an "onset incremental response," defined as the difference between the response to an onset tone at level $I + \delta I$ and one at level I. Two important observations were: 1) The steady-state and onset IR's were nearly equal for stimuli of intermediate range, but the steady-state IR was somewhat larger for stronger stimui, and 2) the ratio of the response R_0 at onset to the response R_τ^- at steady state was approximately equal to 2.5, regardless of the onset intensity level.

This was the most difficult experimental paradigm to match with the model. The rapid AGC and the short-term adaptation circuit tend to impose opposing constraints on the outputs. It was possible to obtain a fairly constant ratio of onset to steady-state response, but this ratio was consistently too large (3.0 instead of 2.5), as shown in Figure 9a. For the parameter settings shown in Table I, the 3-dB onset incremental response of the model was slightly larger than the 3-dB steady-state incremental response for weak signals, but became significantly smaller for stronger signals. This result is in close agreement with the data, as shown in Figure 9b.

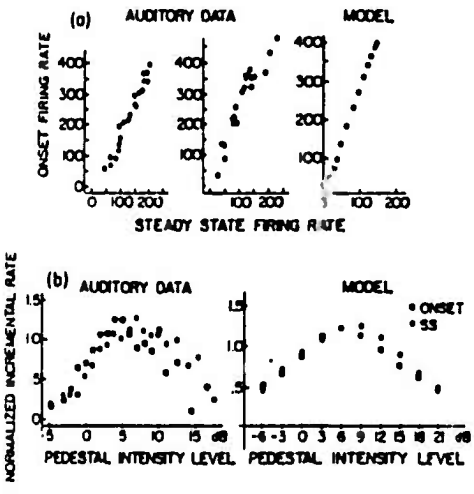


Figure 9: (a) Plots of onset firing rates versus steady-state firing rates, in response to tone pedestals at CF, for two auditory neurons (*left*, from Smith and Zwislocki [2]), and for the computer model (*right*). Model response is for the 2000-Hz channel. (b) (*left*) Plots of median normalized 3-dB incremental responses for 10 auditory neurons (from Smith and Zwislocki [2]) at onsets (open circles) and at steady-state conditions. (*right*) Plots of normalized 3-dB incremental responses for model, at onsets (open circles) and at steady-state conditions.

Synchrony Falloff:

Johnson [8] gave a specific definition for a "synchronization index" that he applied to the period histograms of the steady-state responses of nerve fibers to tone stimuli. This index was defined as

$$S_f = A(F_0)/A(0) \tag{4}$$

where S_f is the synchronization index, $A(f)$ is the amplitude of the spectrum of the period histogram at frequency f, and F_0 is the tone frequency. Johnson measured S_f for a large number of fibers, for tones not necessarily at CF, and obtained the plot shown in Figure 10. Superimposed as large triangles on the plot are points obtained by applying the same definition for synchrony to the model outputs. The main factor controlling the synchrony falloff in the model is the lowpass filter.

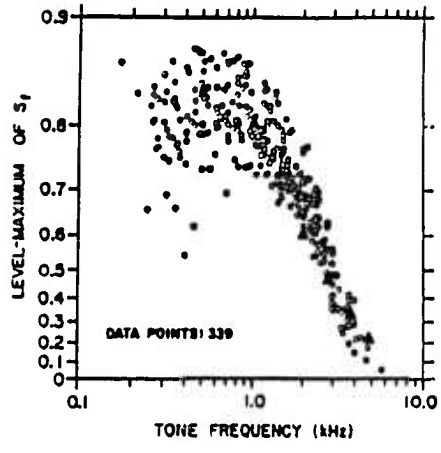


Figure 10: Scatter diagram of synchronization index (from Johnson [8]) as defined in equation 4, as a function of tone frequency (339 measurements from 233 units), with model results superimposed as large triangles.

OUTPUTS OF THE MODEL FOR SPEECH SIGNALS

Figure 11 shows an example of the Stage II outputs for a short segment of a male speaker's voiced speech, during the /e/ of the word "make." Part *a* gives the wideband spectrogram of the word, with a vertical bar indicating the time at which channel outputs are shown in part *b*. The 50 ms time window includes about five pitch periods. The peaks are skewed slightly to the left for low frequencies, a feature that has been observed in auditory data as well [8]. Part *c* of the figure shows the output of the channel whose CF is at F_2 of the vowel. A prominent component at the formant frequency is evident. Such formant periodicity is utilized by the synchrony algorithm in Stage III.

Figure 12 compares Stage I outputs with Stage II outputs for the word "description" spoken by a female speaker. Each waveform is the smoothed output of one of the 40 channels as a function of time, with low-frequency channels at the bottom. It is essential to represent Stage I outputs by a log magnitude rather than a magnitude; otherwise the vowel peaks are overwhelmingly larger than the rest of the data. Log magnitude also corresponds to traditional analysis methods. Because of the saturating nonlinearity in the half-wave rectifier as well as in the final AGC, a log representation is not appropriate for Stage II outputs. The phonetic transcription has been superimposed, to help in judging where segment boundaries should be detected.

All segment boundaries, with the exception of the /ɹ/, are well delineated in the Stage II representation. The closure intervals for both the /k/ and the /p/ are flat valleys in the Stage II representation; there is clear evidence for forward masking here, particularly in the low-frequency region for the /p/. The vowel /ɪ/ has masked low frequency noise not only during the /p/ closure interval but also during the subsequent /ʃ/. Such masking phenomena should enhance the contrast between vowels and fricatives. The boundary between the /ɪ/ and the final /n/ is very difficult to see in the Stage I representation, but there is a much greater hope of detecting it after the Stage II nonlinearities. The stop burst onsets for the /d/ and the /k/ are also much sharper after Stage II.

SUMMARY AND CONCLUSIONS

This paper describes the nonlinear component of a relatively simple model for auditory processing of speech signals, which attains a reasonably good match to measured auditory responses for a number of different experimental paradigms. The model offers the hope of elucidating further the nature of auditory response to speech. In addition, we anticipate that representations obtained from such a model will be well-suited to applications in computer speech recognition.

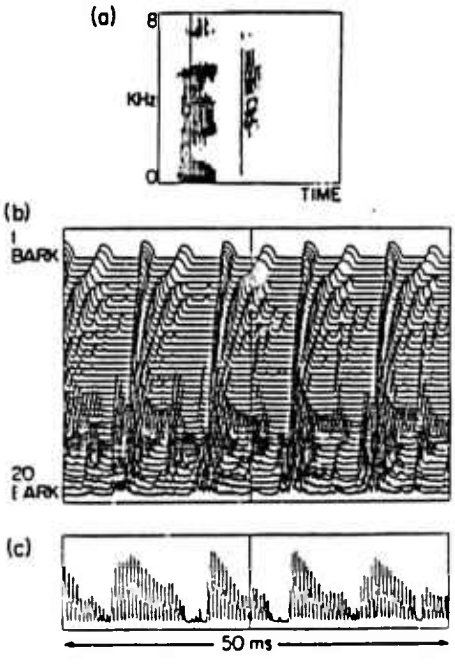


Figure 11: (a) Wideband spectrogram of the word "make," spoken by a male speaker. (b) Stage II outputs of 40 channels, with the *lowest* frequency channel at the *top*, for five pitch periods during the vowel /e/ at the time of the vertical bar in part *a*. (c) Output of the single channel at the frequency of the second formant at the same time as in part *b*.

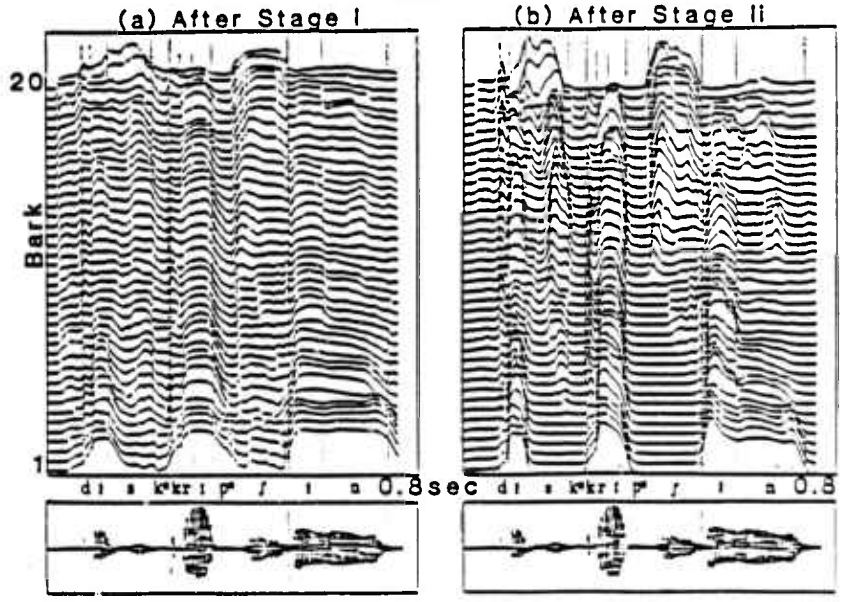


Figure 1 (a) Log magnitude response of Stage I outputs for the word "description" spoken by a female speaker, with the *lowest* frequency channel at the *bottom*. (b) Magnitude response of Stage II outputs for the same word. The phonetic transcription is superimposed on the plots, and the original waveform is shown below in each case.

The Smith and Zwislocki data showing a constant ratio of onset to steady-state response, and a close-to-equal incremental response characteristic for onset and steady-state conditions, have led to the hypothesis that the adaptation process might be linear in nature. A possible alternative explanation, based on the results from the model described here, is that this apparently linear feature may be attributed to a cascade of an enhancing nonlinearity with a compressive nonlinearity, such that the two effectively cancel one another under certain conditions.

The model used for the AGC is a poor approximation of the refractory effect as it is currently understood. First, equation 3 is only valid for steady-state conditions, and only exact for signals that are periodic with Δ. Second, a leaky integrator yields an averaging window for $< x >$ that is exponential in shape, whereas a rectangular window is a much better approximation to the recovery function. Nonetheless, the value for K_{AGC} that was determined experimentally to best match auditory data is .002. This value corresponds to a 2-ms lockout period, which is a little long but at least the correct order of magnitude. Perhaps a more realistic model for the refractory effect that would be appropriate during onsets as well as steady states would result in a better match to the dynamics of the onset envelope response.

It is still premature to suggest that an auditory-based speech analysis system will pay off in speech recognition. There are emerging, however, strong indications that auditory-based representations are interesting and worthy of further study. Onset and offset enhancement properties are particularly effective in sharpening segment boundaries, as discussed in [10]. The forward masking phenomenon should be effective in reducing noise in stop bursts and enhancing low/high frequency contrast in strong fricatives. We are now becoming more confident in the validity of the computer models, such that they may reveal interesting effects in auditory speech processing, which may lead the way to appropriate later-stage speech recognition strategies.

REFERENCES

[1] Goldhor, R. S. (1985) "Representation of consonants in the peripheral auditory system: a modeling study of the correspondence between response properties and phonetic features," RLE Technical Report 505, M.I.T., Cambridge, MA.

[2] Smith, J. C., and J. J. Zwislocki (1975) "Short-term adaptation and incremental responses of single auditory-nerve fibers," *Biological Cybernetics*, 17, 169-182.

[3] Seneff, S. (1986) "A computational model for the peripheral auditory system: application to speech recognition research," *Proceedings of ICASSP-86*, 4, 37.8.1-37.8.4.

[4] Hudspeth, A. J. and D. P. Corey (1977) "Sensitivity, polarity, and conductance change in the response of vertebrate hair cells to controlled mechanical stimuli," *Proceedings of the National Academy of Science U.S.A.*, 74, 2407-2411.

[5] Eggermont, J. J. (1973) "Analog modelling of cochlear adaptation," *Kybernetik*, 14. 117-126.

[6] Johnson, D. H. and A. Swami (1983) "The transmission of signals by auditory-nerve fiber discharge patterns," *Journal of the Acoustical Society of America*, 74(2), 493-501.

[7] Hudspeth, A. J. and D. P. Corey (1977) "Sensitivity, polarity, and conductance change in the response of vertebrate hair cells to controlled mechanical stimuli," *Proceedings of the National Academy of Science U.S.A.*, 74, 2407-2411.

[8] Johnson, D. H. (1974) "The response of single auditory-nerve fibers in the cat to single tones: Synchrony and average discharge rate," Ph.D. Thesis, M.I.T., Cambridge, MA.

[9] Delgutte, B. (1980) "Representation of speech-like sounds in the discharge patterns of auditory-nerve fibers," *Journal of the Acoustical Society of America*, 68, 843-857.

[10] Glass, J. and V. Zue (1987) "Acoustic Segmentation and Classification," these Proceedings.

VOWEL RECOGNITION BASED ON "LINE-FORMANTS" DERIVED FROM AN AUDITORY-BASED SPECTRAL REPRESENTATION*

Stephanie Seneff

Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

A new approach to vowel recognition is described, which begins by reducing a spectrographic representation to a set of straight-line segments that collectively sketch out the formant trajectories. These "line-formants" are used for recognition by scoring their match to a set of histograms of line-formant frequency distributions determined from training data for the 16 vowel categories in the recognition set. Speaker normalization is done by subtracting F_0 from line-formant frequencies on a Bark scale. Although the formants are never enumerated or tracked explicitly, the frequency distributions of the formants are the main features influencing the recognition score. Recognition results are given for 2135 vowels extracted from continuous speech spoken by 292 male and female speakers.

INTRODUCTION

The formant frequencies are probably the most important information leading to the recognition of vowels, as well as other sonorant and even possibly obstruent sounds. Therefore, researchers have spent a considerable amount of effort designing robust formant trackers, which attempt to associate peaks in the spectrum with formant frequencies, using continuity constraints to aid in the tracking of the formants. Once the formant tracks are available, it then becomes possible to identify directions and degree of formant movements, features that are important in recognizing diphthongs, semivowels, and place of articulation of adjacent consonants.

It is impossible to design a "perfect" formant tracker. The most serious problem with formants is that when they are wrong there are often gross errors. Therefore, we have decided to adopt a somewhat different approach, one that can lead to information about formant movements without explicitly labelling the formant numbers. The method also collapses the two stages of formant tracking and track interpretation (e.g., "rising formant") into a single step. The outcome is that a spectrographic representation is reduced to a skeleton sketch consisting of a *set* of straight-line segments, which we call "line-formants," that collectively trace out the formant tracks. The recognition strategy then involves matching all of the line-formants of an unknown segment to a set of templates, each of which describes statistically the appropriate line-formant configurations for a given phonetic class (which could be as detailed as "nasalized /æ/" or as general as "front vowel"). Usually the number of line-formants for a given speech segment is considerably larger than the number of formants, because in many cases several straight-line segments are required to adequately reflect the transitions of a single formant.

*This research was supported by DARPA under Contract N00039-85-C-0254, monitored through Naval Electronic Systems Command.

SIGNAL PROCESSING

Spectral Representation

The system makes use of two spectrogram-like representations that are based on our current understanding of the human auditory system [1]. The analysis system consists of a set of 40 critical band filters, spanning the frequency range from 160 to 6400 Hz. The filter outputs are processed through a nonlinearity stage that introduces such effects as onset enhancement, saturation and forward masking. This stage is described in detail in a companion paper [2]. The outputs of this stage are processed through two independent analyses, each of which produces a spectrogram-like output. The "Mean Rate Spectrogram" is related to mean rate response in the auditory system, and is used for locating sonorant regions in the speech signal. The "Synchrony Spectrogram" takes advantage of the phase-locking property of auditory nerve fibers. It produces spectra that tend to be amplitude-normalized, with prominent peaks at the formant frequencies. The amplitude of each spectral peak is related to the amount of energy at that frequency *relative* to the energy in the spectral vicinity. The line-formant representation is derived from this Synchrony Spectrogram.

Line-formant Processing

The line-formants are obtained by first locating sonorant regions, based on the amount of low frequency energy in the Mean Rate Spectrogram. Within these sonorant regions, a subset of robust peaks in the Synchrony Spectrogram is selected. Peaks are rejected if their amplitude is not sufficiently greater than the average amplitude in the surrounding time-frequency field. For each selected peak, a short fixed-length line segment is determined, whose direction gives the best orientation for a proposed formant track passing through that peak, using a procedure as outlined in Figure 1. The amplitude at each point on a rectangular grid within a circular region surrounding the peak in question is used to update a histogram of amplitude as a function of the angle, θ. Typical sizes for the circle radius are 20 ms in time and 1.2 Bark in frequency. The maximum value in the histogram defines the amplitude and corresponding θ for the proposed track, as marked by an arrow in Figure 1*c*.

At each time frame several new short segments are generated, one for each robust spectral peak. A short segment is then merged with a pre-existing partial line-formant whenever the two lines have a similar orientation, and the distance between each endpoint and the other line is sufficiently small. The merging process is accomplished by creating a weighted-average line-formant that incorporates the new line. If a given new segment is sufficiently unique, it is entered as a new partial line-formant.

The resulting *Skeleton Spectrogram* for the /ɑ/ in the word "shock" is illustrated in Figure 2*a*, along with a *Schematized Spectrogram* in Figure 2*b*, included to facilitate visual evaluation. The latter is constructed by replacing each line-formant with a time sequence of Gaussian-shaped spectral peaks with amplitude equal to the line's amplitude. The corresponding Synchrony Spectrogram is shown in Figure 2*c*, with line-formants superimposed. For direct comparison, Figure 2*d* shows a Synchrony Spectral cross section at the time of the vertical bar, on which is superimposed a cross section of the Schematized Spectrogram. For this example, we see that peak locations and amlitudes in the vowel are accurately reflected. In addition, formant transitions appropriate for the palatal fricative on the left and the velar stop on the right are also captured.

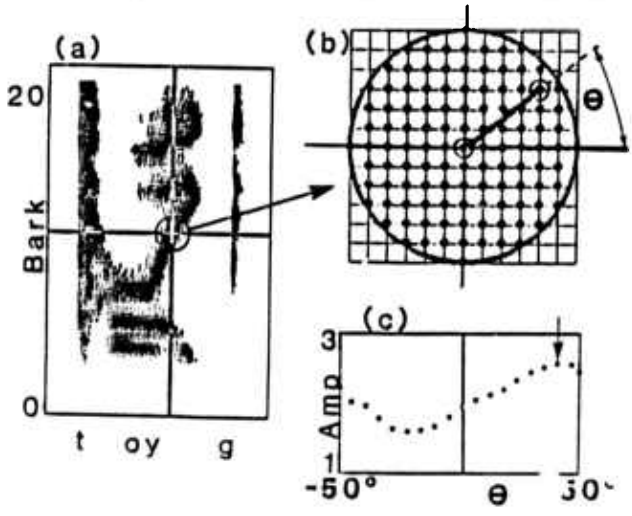


Figure 1: Schematic illustration of process used to determine an orientation for a formant passing through a peak. (a) Synchrony Spectrogram with cross-bars indicating a referenced peak. (b) Schematic blow-up of region around the peak, outlining procedure to generate a histogram of amplitude as a function of angle. (c) Resulting histogram for the example in part *a*.

RECOGNITION EXPERIMENT

Thus far, we have focused our studies on speaker-independent recognition for 16 vowels and diphthongs of American English in continuous speech, restricted to obstruent and nasal context. The semivowel context is excluded because we believe that in many cases vowel-semivowel sequences should be treated as a single phonetic unit much like a diphthong.

Speaker Normalization

Our first task was to devise an effective speaker-normalization procedure. Many investigators have noted the strong correlation between formant frequencies and F_0 [3]. The relationship is clearly nonlinear – the second formant

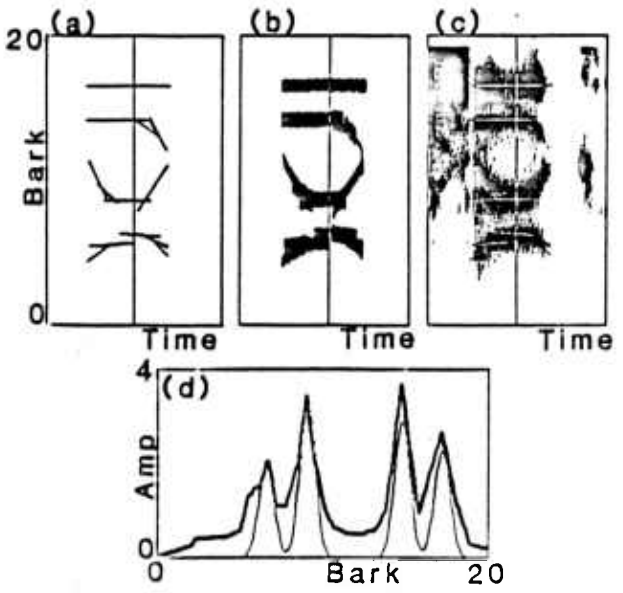


Figure 2: Sample line-formant outputs: (a) Skeleton Spectrogram for word "shock," (b) Corresponding Schematized Spectrogram, (c) Synchrony Spectrogram with line-formants superimposed, (d) cross-sections from *b* and *c* at the cursor, superimposed.

for female /i/ is higher on average by several hundred Hz, whereas the F_0 difference is on the order of 100 Hz. However, on a Bark (critical band) scale the male-female difference in F_2 for /i/ becomes much more similar to that in F_0. Thus we decided to try a very simple scheme – for each line-formant, subtract from the line's center frequency the *median* F_0 over the duration of the line, on a Bark scale.

We found this normalization procedure to be remarkably effective, as illustrated in Figure 3. Part *a* shows a histogram of the center frequencies of all of the lines for 35 male and 35 female /æ/ tokens. Part *b* shows the same data, after median F_0 has been subtracted from each line's center frequency. The higher formants emerge as separate entities after the F_0 normalization. The normalization is not as effective for F_1, but the dispersal in F_1 is due in part to other factors such as vowel nasalization.

A valid question to ask is the following: if it is supposed that speaker normalization can be accomplished by subtracting *a factor times* F_0 from all formant frequencies, then what should be the numerical value of the factor? An answer can be obtained experimentally using autoregressive analysis. We defined $F_n' = F_n - \alpha F_0$ to be the normalized formant frequency for each line. Using vowels for which the formants are well separated, we associated a group of lines with a particular formant such as F_2. The goal was to minimize total squared error for each remapped formant among all speakers, with respect to α. The resulting estimated value for α was 0.975, providing experimental evidence for the validity of the proposed scheme.

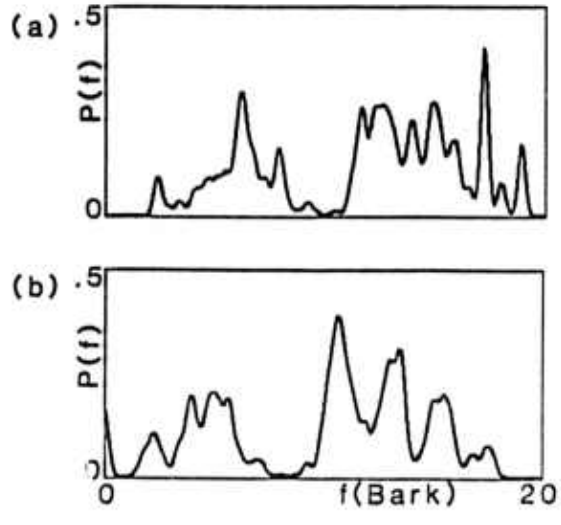


Figure 3: Histograms for center frequencies of all line-formants for 35 female and 35 male tokens of /æ/, (a) without F_0 normalization, and (b) with F_0 normalization.

Scoring Procedures

Our goal in developing a recognizer for the vowels was to emphasize the formant frequency information without ever explicitly identifying the formant numbers. We wanted to avoid traditional spectral template-matching schemes, because they depend too heavily on irrelevant factors such as the loudness or the overall spectral tilt. On the other hand, we did not want to specify, for example, the distance between F_2 and a target F_2, because this relies on accurately enumerating the formants.

We decided to construct histograms of frequency distributions of spectral peaks across time, based on data derived from the line-formants. The scoring amounts to treating each histogram as a probability distribution, and matching the unknown token's line-formants against the appropriate distributions for each vowel. To construct the histograms for a given vowel, all of the line-formants in a training set were used to generate *five* histograms intended to capture the distributions of the formants at significant time points in the vowel. All lines were normalized with respect to F_0, which was computed automatically using a version of the Gold-Rabiner pitch detector [4]. Each line-formant's contributions to the histograms were weighted by its amplitude and its length.

Only left, center and right frequencies of the lines were used in the histograms. The left frequency of a given line-formant falls into one of two bins, depending upon whether or not it is near the beginning of the vowel. Right frequencies are sorted similarly, with a dividing point near the end of the vowel. Center frequencies are collected into the same histogram regardless of their time location. Such a sorting process results in a set of histograms that reflects general formant motions over time. For example, the F_2 peak in the histograms for /e/ shifts upward from left-on-left to center to right-on-right, reflecting the fact that /e/ is diphthongized towards a /y/ off-glide, as illustrated in Figure 4.

To score an unknown token, the left, center, and right frequencies of all of its lines are matched against the appropriate histograms for each vowel category, which are treated as probability distributions. The score for the token's match is the weighted sum of the log probabilites for the five categories for all of the line-formants. The amplitude of the line does not enter into the match, but is used only as a weight for the line's contribution to the score. This strategy eliminates the problem of mismatch due to fact such as spectral tilt or overall energy.

Recognition Results

The vowels used for recognition were extracted from sentences in the TIMIT database [5]. The speakers represented a wide range of dialectical variations. A total of 2135 vowel tokens spoken by 206 male and 82 female

2135 Vowels, 288 Speakers

u	i	ɪ	e	ɛ	æ	ɑ˞	ɔɪ	ɑ	ʌ	ɔ	ɔɪ	o	ʊ	u	ɝ
90	220	268	128	183	158	131	92	103	147	156	96	83	114	96	103

Table 1: Distributions of vowels in recognition experiment

speakers were used as both training and test data, using a jackknifing procedure. The distributions of vowels are shown in Table 1. Each speaker's vowel tokens were scored against histograms computed from all of the line-formants *except* those from that speaker. The scoring procedure was as discussed above, with histograms defined for sixteen vowel categories. The endpoints for the vowels were taken from the time-aligned phonetic transcription.

A matrix of first-choice confusion probabilities is given in Table 2, in terms of percent correct in the phonetic category. For the most part, confusions are reasonable. We feel encouraged by this performance, especially considering that multiple dialects and multiple contexts are included in the same histogram.

Figure 5 summarizes recognition performance in terms of percentage of time the correct answer is in the top N, for all speakers, and for male and female speakers separately. Recognition was somewhat worse for females, who represented only 25% of the population. Also shown are the recognition results for female speakers when the F_0-normalization scheme is omitted, both in collecting the histograms and in scoring. Significant gains were realised as a consequence of the normalization. The performance for the male speakers without F_0 normalization however (not shown) did not change.

	u	i	ɪ	e	ɛ	æ	ɑ˞	ɔɪ	ɑ	ʌ	ɔ	ɔɪ	o	ʊ	u	ɝ
u	49	15	11	6	3	1								4	8	3
i	11	70	5	8	1					1			1	2	1	
ɪ	11	10	32	16	11	5				2				6	5	2
e	5	9	5	60	7	6	1	1		1		3		2		
ɛ	3	1	12	14	37	16	1	1		6			2	3	1	4
æ	1	1	1	10	9	59	4	7		4		1		1	2	1
ɑ˞					2	13	39	6	13	7	7	2	7	3		2
ɑɪ					1	7	4	58	15	2	1	8	2	2		
ɑ						3	7	15	40	4	27		5			
ʌ	1	1	1		7	6	2	5	17	39	2	2	5	6	5	1
ɔ							3	3	29	1	46	4	12	1	1	1
ɔɪ							1	1	10	1		3	67	8		6	2
o							4	[illegible]	1	5	6	14	4	53		5	1
ʊ	20	1	11		4	1			1	5	3	3	9	28	11	4
u	11	2	2	2	1	1	1	1	2	4	1	2	9	17	40	3
ɝ	8		4	3	3		1	1	1	3	1	1	5	5	3	62

Table 2: First choice confusion matrix for the vowels
Row = Labeled Category, Column = Recognized Category.

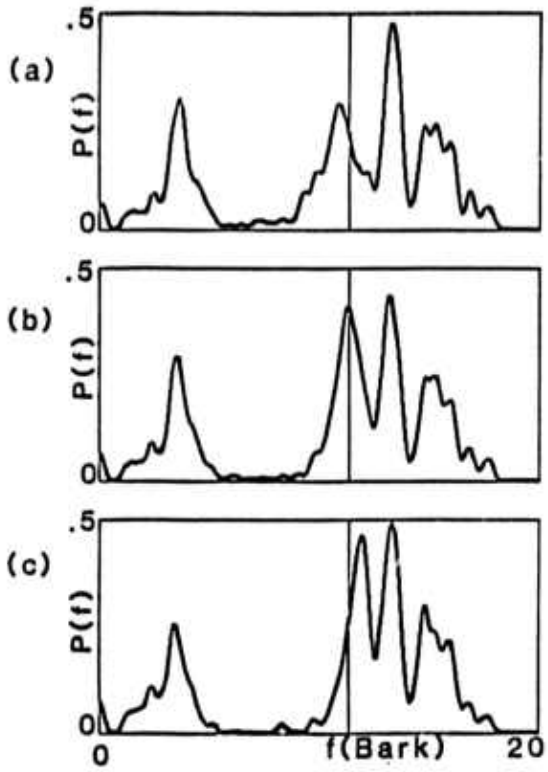


Figure 4: Histograms for (a) left-on-left, (b) center, and (c) right-on-right line-formant frequencies for 128 tokens of /e/, F_0 normalized.

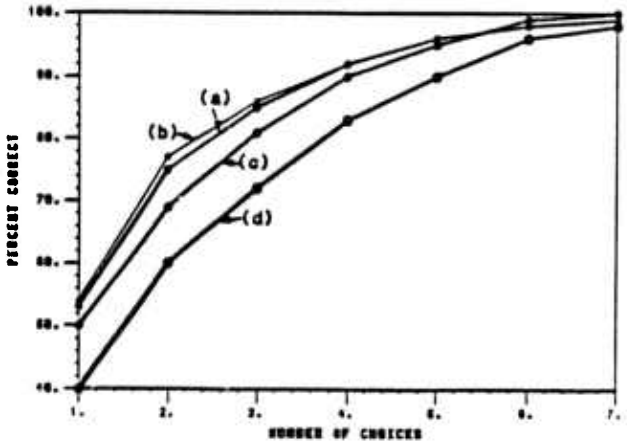


Figure 5: Recognition results expressed as percent of time correct choice is in top N, for the following conditions: (a) all speakers, (b) males only, (c) females only, and (d) females without F_0 normalization.

FUTURE PLANS

We believe that recognition performance can be improved by extensions in several directions. One is to divide each vowel's histograms into multiple subcategories, based on both general features of the vowel and coarticulation effects. General categories, useful for the center-frequency histogram, would include "nasalized," "Southern accent," or "fronted." Left- and right-context place of articulation, such as "velar," could be used to define corresponding histogram subcategories. We also plan to explore an alternative recognition strategy for explicitly matching each *line-formant* against a set of template *line-formants* describing a particular phonetic category, instead of reducing the line to three "independent" points. We believe that such an approach will better capture the fact that a given left frequency and a given right frequency are connected. Finally, we plan to gradually expand the scope of the recognizer, first to vowels in all contexts and then to other classes such as semivowels.

REFERENCES

[1] Seneff, S. (1986) "A Computational Model for the Peripheral Auditory System: Application to Speech Recognition Research," ICASSP Proceedings, Tokyo, Japan, 37.8.1-37.8.4.

[2] Seneff, S. (1987) "A New Model for the Transduction Stage of the Auditory Periphery," (these proceedings).

[3] Syrdal, A. K. (1985) "Aspects of a Model of the Auditory Representation of American English Vowels," Speech Communication **4**, 121-135.

[4] Gold, B. and L.R. Rabiner (1969) "Parallel Processing Techniques for Estimating Pitch Periods of Speech in the Time Domain," J. Acoust. Soc. Am. **46**, 442-448.

[5] Lamel, L. F., R. H. Kassel, and S. Seneff (1986) "Speech Database Development: Design and Analysis of the Acoustic-Phonetic Corpus," Proceedings of the DARPA Speech Recognition Workshop Palo Alto, CA., Feb 19-20, 100-109.

ACOUSTIC SEGMENTATION AND CLASSIFICATION*

James R. Glass and Victor W. Zue

Department of Electrical Engineering and Computer Science
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

As part of our goal to better understand the relationship between the speech signal and the underlying phonemic representation, we have developed a procedure that describes the acoustic structure of the signal, and have determined an acoustically motivated set of broad classes. Acoustic events are embedded in a multi-level structure, in which information ranging from coarse to fine is represented in an organized fashion. An analysis of the acoustic structure, using 500 utterances from 100 different talkers, shows that it captures over 94% of the acoustic-phonetic events of interest with an insertion rate of less than 8%. Acoustic classification is accomplished using a hierarchical clustering technique. Our evaluations of the results show that with a small number of clusters, we are able to obtain a robust description of the speech signal and to provide a meaningful acoustic-phonetic interpretation.

INTRODUCTION

The task of phonetic recognition can be stated broadly as the determination of a mapping of the acoustic signal to a set of phonological units (e.g., distinctive feature bundles, phonemes, or syllables) used to represent the lexicon. In order to perform such a mapping, it is often desirable to first transform the *continuous* speech signal into a *discrete* set of segments. Typically, this *segmentation* process is followed by a *labeling* process, in which the segments are assigned phonetic labels. While this procedure is conceptually straightforward, its implementation has proved to be immensely difficult [4]. Our inability to achieve high-performance phonetic recognition is largely due to the diversity in the acoustic properties of speech sounds. Stop consonants, for example, are produced with abrupt changes in the vocal tract configuration, resulting in distinct acoustic landmarks. Semivowels, on the other hand, are produced with considerably slower articulatory movements, and the associated acoustic transitions are often quite obscure. To complicate matters further, the acoustic properties of phonemes change as a function of context, and the nature of such contextual variation is still poorly understood. As a result, the development of algorithms to locate and classify these phonemes-in-context, or allophones, typically requires intense knowledge engineering.

We are presently exploring a somewhat different approach to phonetic recognition in which the traditional phonetic-level description is bypassed in favor of directly relating the acoustic realizations to the underlying phonemic forms. Our approach is motivated by the observation that a description based on allophones is both incomplete and somewhat arbitrary. Phoneticians traditionally identify a certain number of important allophones for a given phoneme based on their examination of a limited amount of data together with introspective reasoning. With the availability of a large body of data [5], we are now in a position to ascertain whether these categories are acoustically meaningful, and whether additional categories will emerge. Rather than describing the acoustic variations in terms of a set of preconceived units, i.e. allophones, we would like to let the data help us *discover* important regularities. In this line of investigation, the speech signal is transformed into a set of *acoustic* segments, and the relationship between these acoustic segments and the underlying phonemic form is described by a grammar which will be determined through a set of training data.

This paper describes some recent work in acoustic segmentation and classification, as part of the development of a phonetic recognition system. Ideally, we would like our system to have the following set of properties. The segmentation algorithm should be able to reliably detect abrupt acoustic events such as a stop burst and gradual events such as a vowel to semivowel transition. More importantly, there must exist a coherent framework in which acoustic changes from coarse to fine can be expressed. The classification algorithm should produce an accurate description of the acoustic events. Phonemes that are acoustically similar should fall into the same class. If, on the other hand, a phoneme falls into more than one acoustic class, then the different acoustic realizations should suggest the presence of important contextual variations.

*This research was supported by DARPA under Contract N00039-85-C-0254, monitored through Naval Electronic Systems Command.

ACOUSTIC SEGMENTATION

The purpose of our acoustic segmentation is to delineate the speech signal into segments that are acoustically homogeneous. Realizing the need to describe varying degrees of acoustic similarity, we have adopted a multi-level representation in which segmentations of different sensitivities are structured in an organized fashion.

Determining Acoustic Segments

The algorithm used to establish acoustic segments is a simplified version of the one we developed to detect nasal consonants in continuous speech [2]. This algorithm adopts the strategy of measuring the similarity of each frame to its near neighbors. Similarity is computed by measuring the Euclidean distance between the spectral vector of a given frame and the two frames 10 ms away. Moving on a frame-by-frame basis from left to right, the algorithm associates each frame in the direction, past or future, in which the similarity is greater. Acoustic boundaries are marked whenever the association direction switches from past to future. By varying the parameters of this procedure, we are able to control its sensitivity in detecting acoustic segments in the speech signal. We have chosen to operate with a low deletion rate because mechanisms exist for us to combine segments if necessary at a later stage.

Signal Representation

The algorithms for both acoustic segmentation and classification use the output of an auditory model proposed by Seneff [7]. The model incorporates known properties of the human auditory system, such as critical-band filtering, half-wave rectification, adaptation, saturation, forward masking, spontaneous response, and synchrony detection. The model consists of 40 filters equally spaced on a Bark frequency scale, spanning a frequency range from 130 to 6,400 Hz. For our application, we use the output of the filter channels after they have been processed through a hair-cell/synapse transduction stage. The envelope of the resulting channel outputs corresponds to the "mean rate response" of the auditory nerve fibers. The outputs are represented as a 40-dimensional feature vector, computed once every 5 ms.

We find this representation desirable for several reasons. The transduction stage tends to enhance the onsets and offsets in the critical-band channel outputs. Forward masking will greatly attenuate many low low-amplitude sounds because the output falls below the spontaneous firing rate of the nerve fibers. These two effects combine to sharpen acoustic boundaries in the speech signal. Furthermore, due to the saturation phenomena, formants in the envelope response appear as broad-band peaks, obscuring detailed differences among similar sounds, an effect we believe to be advantageous for grouping similar sounds. In a series of experiments comparing various signal representations for acoustic segmentation, we found that, over a wide range of segmentation sensitivities, the auditory-based representation consistently produced the least number of insertion and deletion errors [3].

Multi-Level Description

Our first experience with acoustic segmentation led us to the conclusion that there exists no single level of segmental representation that can adequately describe all the acoustic events of interest. As a result, we have adopted a multi-level representation similar to the scale-space proposal by Witkin [9]. We find this representation attractive because it is able to capture both coarse and fine information in one uniform structure. Acoustic-phonetic analysis can then be formulated as a path finding problem in a highly constrained search space.

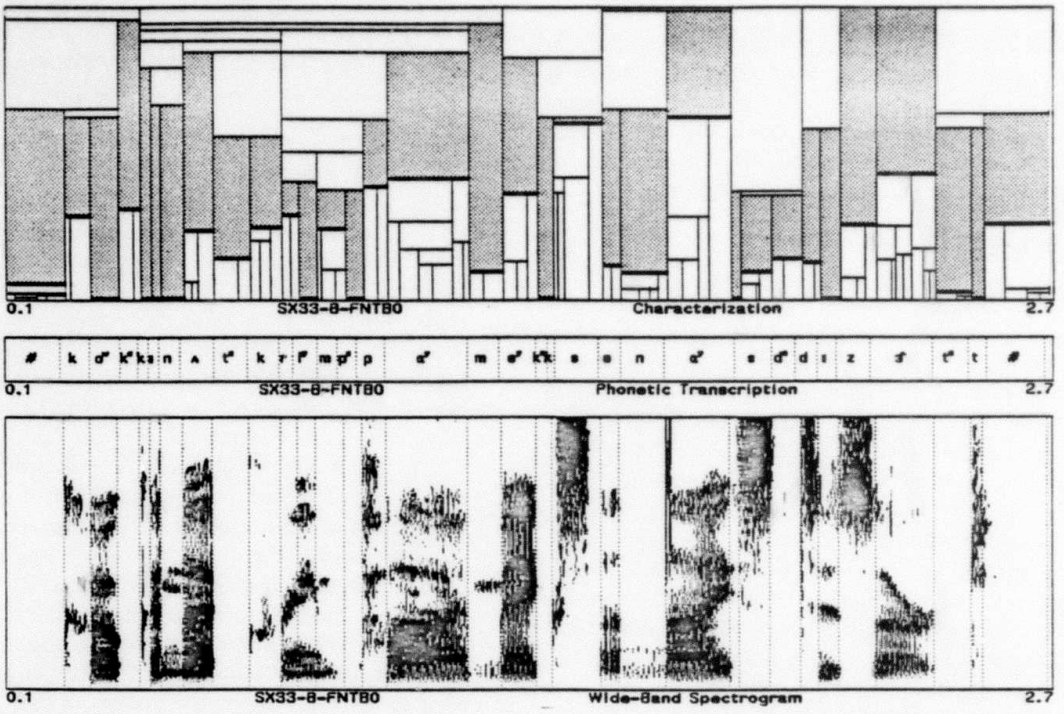
The procedure for obtaining a multi-level representation is similar to that used for finding acoustic segments. First, the algorithm uses all of the proposed segments as "seed regions". Next, each region is associated with either its left or right neighbor using a similarity measure. When two adjacent regions associate with each other, they are merged together to form a single region. Similarity is computed with a weighted Euclidean distance measure applied to the average spectral vectors of each region. This new region subsequently associates itself with one of its neighbors. The merging process continues until the entire utterance is described by a single acoustic event. By keeping track of the distance at which two regions merge into one, the multi-level description can be displayed in a tree-like fashion as a dendrogram, as illustrated in Figure 1 for the utterance "Coconut cream pie makes a nice dessert". From the bottom towards the top of the dendrogram the acoustic description varies from fine to coarse. The release of the initial /k/, for example, may be considered to be a single acoustic event or a combination of two events (release plus aspiration) depending on the level of detail desired.

We found this procedure to be more attractive than the scale-space representation which we and others have investigated [6,8]. The scale-space procedure produces a multi-level description by uniformly increasing the scale through lowpass filtering, without regard to local context. As a result, at low scales it tends to eliminate short but distinct acoustic events such as stop releases and flaps. In contrast, our procedure merges regions using a local similarity measure. As a result, regions that are acoustically distinct are typically preserved higher in the dendrogram, regardless of their duration. Finally, by representing each region by a single average spectral vector, our procedure is computationally more efficient.

Evaluation

We have evaluated the effectiveness of our multi-level acoustic representation in several ways. First, we devel-

Figure 1: Multi-level Acoustic Segmentation.



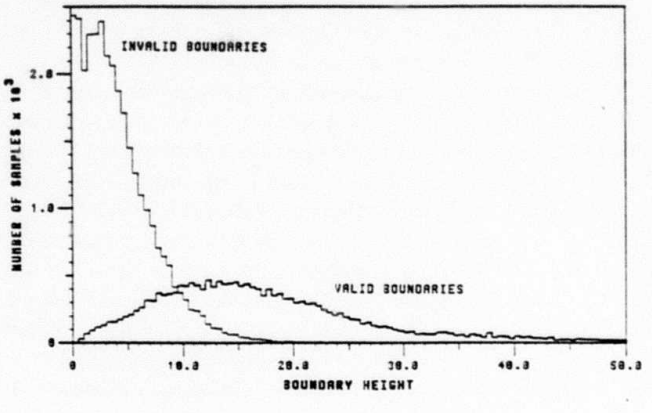
oped an algorithm to automatically find the path through the dendrogram which best matched a time-aligned phonetic transcription. An example of such a path is highlighted on the dendrogram in Figure 1. The boundaries along this path are also marked by vertical lines in the spectrogram. We then tabulated the insertion and deletion errors of these paths. Not only should we expect a small number of insertion and deletion errors, the errors should also be acoustically reasonable. Next, we compared the time difference between the boundaries found and the actual boundaries as provided by the transcriptions. Finally, we examined whether correct and incorrect boundaries behave in any reasonable way.

The evaluation was carried out using 500 sentences from the TIMIT database [5]: five sentences each from 100 talkers (69 male and 31 female). These sentences contained nearly 18,500 phones. The best-path alignment procedure gave under 6% and 8% deletion and insertion errors, respectively. Closer examination of the errors reveals that the deletions mostly involve acoustic transitions that are not always distinct, such as those between closures and weak stop releases, between vowels and semivowels, between nasals and voiced closures, and between stops and fricatives. In Figure 1, we can see that the boundary between the stop and the fricative was deleted in the word 'makes'. We have not yet analyzed the insertions as exhaustively as the deletions. However, it appears that approximately half of the insertions occur within the boundaries of a vowel. In Figure 1 there was an insertion between the vowel and the fricative in the word 'nice'.

Analysis of the time difference between the boundaries found and those provided by the transcription shows that that more than 70% of the boundaries were within 10 ms of each other, and more than 90% were within 20 ms.

Finally, we compared the boundary heights in the dendrogram (as measured by the distance at which the region is merged with one of its neighbors) of valid boundaries to those of invalid boundaries. This comparison is shown in Figure 2. The valid boundaries are typically higher, suggesting that they are more resilient against merging.

Figure 2: Histogram of Boundary Height.



ACOUSTIC CLASSIFICATION

Once a signal has been segmented, each region in the dendrogram is assigned an acoustic label using a pattern classification procedure described in this section. Ideally the classification procedure should group similar speech sounds into the same category, and separate sounds that are widely different. While we did not know how many classes would be appropriate, we suspected that the number of classes would be small so that the results would be robust against contextual and extra-linguistic variations.

Hierarchical Classification

In order to classify the acoustic segments, we first determined a set of prototype spectral templates based on training data. In our case this was accomplished by using a stepwise-optimal hierarchical clustering procedure [1]. This technique, which is conceptually simple, structures the data explicitly. In addition, the number of clusters need not be specified in advance. We used an agglomerative, or bottom-up procedure because the merging criterion is easier to define than the splitting criterion needed for the divisive procedure. The agglomerative technique is also computationally less demanding than the divisive technique. Pilot studies performed using several databases containing many talkers indicated that the agglomerative procedure produced relatively stable results, provided sufficient training data is available.

In the interest of reducing the amount of necessary computation, we took several steps to reduce the size of the training sample used in the hierarchical clustering procedure. First, all of the frames within a dendrogram region were represented by a single average spectral vector. Second, rather than using all regions in the dendrogram for training, we included only those regions that the path finding algorithm had used for alignment with the phonetic transcription. These two steps were found experimentally to reduce the data by a factor of fifteen, with no noticeable degradation in clustering performance.

Further data reduction was achieved by merging similar spectral vectors with an iterative nearest neighbor procedure, in which a vector is merged into an existing cluster if the distance between it and the cluster falls below a threshold. Otherwise, a new cluster is formed with this vector, and the procedure repeats. In the end, all clusters with membership of two or less are discarded, and the data are resorted. The value of the threshold was determined experimentally from a subset of the training data, and was set to maximize the number of clusters with more than two members. This final step was found experimentally to reduce the size of the data by a factor of thirty.

We used the same 500 TIMIT sentences to train the classifier. These data comprised over 24 minutes of speech and contained over 290,000 spectral frames. Restriction to the time-aligned dendrogram regions reduced the data to just under 19,000 regions. The pre-clustering procedure on these regions produced 560 clusters covering nearly 96% of the original data. All of the data then were resorted into these 560 seed clusters. The distribution of the cluster sizes was a well-behaved exponential function with a mode of 6, median of 16, and an average size of 34. The hierarchical clustering was then performed on these seed clusters.

Cluster Evaluation

The hierarchical clustering algorithm arranges the clusters in a tree-like structure in which each node bifurcates at a different level. The experimenter thus has the freedom to select the number of clusters and the associated spectral templates for pattern classification. We have performed several types of analysis to help us make this decision.

First, the set of clusters should be acoustically robust. By performing the clustering experiment on several databases and examining the phonetic contents of the clusters, we observed that the top three or four levels of the tree structure are quite stable. For instance, the top two clusters essentially separate all consonants from vowels. The vowel cluster subsequently divides based on spectral shapes corresponding to different corners of the vowel triangle. The obstruent cluster divides into subgroups such as silence, nasals, and fricatives. From these observations we decided that the number of clusters for reliable pattern classification should not exceed twenty.

We also measured the average amount of distortion involved in sorting the training set into a given set of clusters. For a given number of clusters, the set with the minimum average distortion was designated as the best representation of the data. Figure 3 illustrates the rate of decrease in the average distortion as the number of clusters increases from one to twenty. From this plot we see that the most significant reductions in the average distortion occur within approximately the first ten clusters. Afterwards the rate of decrease levels off to around 1%.

Figure 3: Average Distortion versus Number of Clusters.

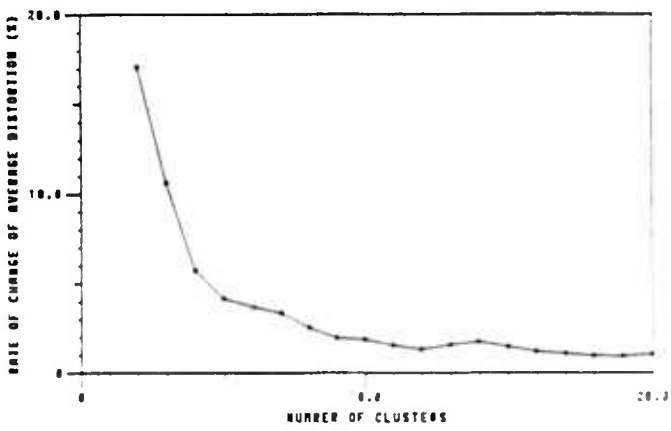
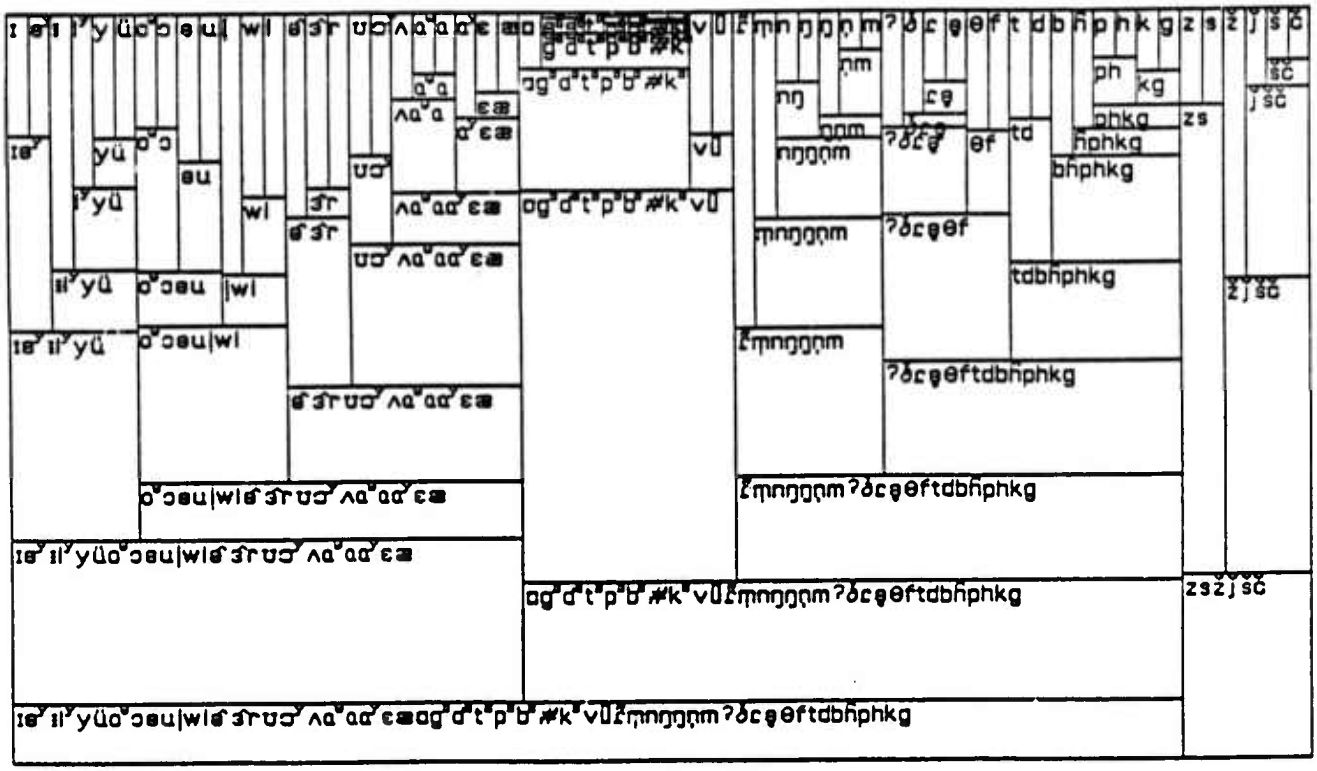


Figure 4: Phonetic Hierarchical Structure with Ten Clusters.

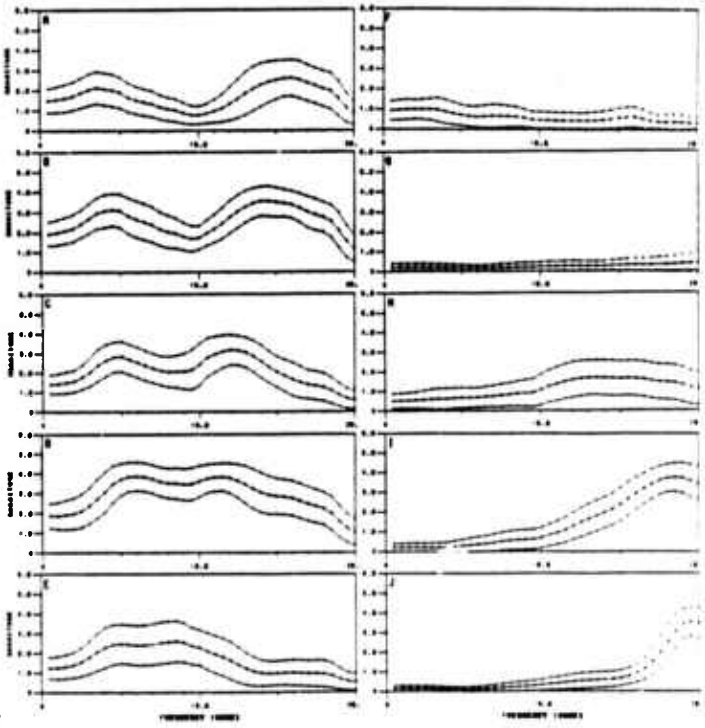


We also judged the relative merit of a set of clusters by examining the distribution of phonetic information within each set. This was done by performing hierarchical clustering of all phones using their distribution across the set of clusters as a feature vector. This procedure is very helpful in facilitating visualization of the data structure captured by a set of clusters. A qualitative analysis of these structures showed that after ten clusters the hierarchical organization did not change significantly. The structure for ten clusters is shown in Figure 4.

Finally, we compared the resulting phonetic distribution for the clusters obtained from the training data to that from a new set of 500 sentences spoken by 100 new speakers. We found that the percentage difference for a given cluster and phoneme is, on the average, around 1%, suggesting that the results did not change significantly. Closer examination reveals that the larger differences are mostly due to sparse data.

Based on the results of these analyses, we concluded that, by selecting approximately ten clusters, we are able to capture a large amount of the variability in the data, as well as a large amount of phonetic information. Furthermore, because the number of clusters is fairly small, we are more confident that this result is acoustically robust in the face of contextual and extra-linguistic effects. The ten clusters produced by the clustering experiment are illustrated in Figure 5.

Figure 5: Spectra of Ten Clusters Illustrating the Average and Deviation.



DISCUSSION

Acoustic Segmentation

The segmentation algorithm uses relational information within a local context. As a result, we believe that it is fairly insensitive to extra-linguistic factors such as recording conditions, spectral tilt, long term amplitude changes, and background noise. Because these procedures require no training of any kind they are also totally speaker-independent. In the future, we plan to examine in more detail the behavior of this algorithm under varying recording conditions.

The results of our experiment on acoustic segmentation suggest that a multi-level representation is potentially very useful. The combined segment insertion and deletion rate of 14% is much better than the best result we were able to obtained previously (25%) with a single-level representation, using essentially the same segmentation algorithm and signal representation [3]. Analysis of the errors indicates that most of the deletions occur when the acoustic change is subtle. When a boundary is inserted, it is often the case that significant acoustic change exists, such as within a diphthong or between the frication and aspiration phases of stop releases. Since our objective is to provide an accurate acoustic description of the signal, some of these insertions and deletions perhaps should not be counted as errors.

The dendrogram produces valid boundaries as well as invalid ones, and the distributions of the heights for these two kinds of boundaries are well separated, as shown in Figure 2. The separation becomes even more pronounced when the distributions are conditioned on the general context of the boundary. This type of information lends itself naturally to a probabilistic framework for finding the best path through the dendrogram.

Acoustic Classification

We are also very encouraged by the results of our acoustic classification procedure. It appears that we can reliably assign each segment to one of a small set of acoustic categories, each having a meaningful phonemic distribution. In other words, phonemes that are acoustically similar by and large fall into the same acoustic class. As a result, we believe that these acoustic labels can help us discover the relationship between phonemes and their acoustic realizations. For example, we found that the phoneme /ð/ predominantly falls into acoustic categories F and H shown in Figure 5. We plan to examine these data more closely to try to understand the context in which each of these realizations is preferred.

The phonetic structure we obtained from our results is also attractive because it provides a totally acoustic motivation for a set of broad classes. Previous research on lexical constraints has shown that knowledge of the broad phonetic categories of the phonemes can be extremely helpful in eliminating unlikely lexical candidates [10]. We believe that the set of acoustic labels that we have determined can potentially aid in the recognition of phonemic classes.

SUMMARY

In summary, we have reported some initial work with acoustic segmentation and classification which we believe can provide a foundation for an eventual phonetic recognition system. By representing the speech signal with a multi-level acoustic description, we are able to capture, and to organize in a meaningful fashion, the majority of acoustic-phonetic events of interest. Our work with acoustic classification indicates that, with a small number of spectral templates, we are able to obtain a robust description of the speech signal, and also to provide a meaningful phonetic interpretation. In the future we will combine these two results and begin to describe in more detail the relationship between the acoustic signal and the underlying phonemic representation.

REFERENCES

[1] Duda, R.O., Hart, P.E., *Pattern Classification and Scene Analysis*, New York, John Wiley and Sons, 1973.

[2] Glass, J.R., Zue, V.W., "Recognition of Nasal Consonants in American English," *Proc. DARPA Speech Recognition Workshop*, Report No. SAIC-86/1546, February 1986.

[3] Glass, J.R., Zue, V.W., "Signal Representation for Acoustic Segmentation," *Proc. of the First Australian Conference on Speech Science and Technology*, November 1986.

[4] Klatt, D.H., "Review of the ARPA Speech Understanding Project," *J. Acoust. Soc. Amer.*, Vol. 62, No. 6, pp. 1345-1366, Dec. 1977.

[5] Lamel, L, Kassel, R., Seneff, S., "Speech Database Development: Design and Analysis of the Acoustic-Phonetic Corpus," *Proc. DARPA Speech Recognition Workshop*, Report No. SAIC-86/1546, February 1986.

[6] Lyon, R.F., "Speech Recognition in Scale Space," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1987.

[7] Seneff, S, "A New Model for the Transduction Stage of the Auditory Periphery," (these proceedings).

[8] Withgott, M., Bagley, S.C., Lyon, R.F., Bush, M.A., "Acoustic-Phonetic Segment Classification and Scale-Space Filtering," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1987.

[9] Witkin, A.P., "Scale Space Filtering: A New Approach to Multi-Scale Description," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 1984.

[10] Zue, V.W., "The Use of Speech Knowledge in Automatic Speech Recognition," *Proc. IEEE*, 1985, vol. 73, no. 11, 1602-1615.

RULE: A SYSTEM FOR CONSTRUCTING RECOGNITION LEXICONS

Mitchel Weintraub and Jared Bernstein

Speech Research Program
SRI International
Menlo Park, CA 94025

ABSTRACT

The RULE software system is a series of tools that allows one to construct recognition lexicons. The tools run on a Symbolics 3600 computer, and allow a user to:

(1) Easily construct and test linguistic rules
(2) Automatically compile and apply rules to probabilistic networks
(3) Graphically display pronunciation networks of words or sentences
(4) Observe the pronunciation networks as they are modified by the linguistic rules
(5) Transcribe speech by selecting one of the possible paths through a pronunciation network
(6) Test if a set of phonological rules can explain observed forms

A previous paper [Bernstein et.al. 1986 DARPA Speech Recognition Workshop] described an earlier version of these tools. This paper describes new algorithms that apply phonological rules to pronunciation networks. Significant recent developments in RULE include: (1) phonological rules are applied to probabilistic pronunciation networks, and (2) generation of interword phonological effects when phonological rules are applied to individual word models in a lexicon.

1. INTRODUCTION

The object of this research is to construct recognition lexicons that can be used in a speaker-independent continuous-speech recognition system. Each word in the vocabulary is modeled by a separate probabilistic pronunciation network. The set of all pronunciation networks, and the algorithms that determine which paths of one pronunciation network can follow which paths of a different pronunciation network constitute the recognition lexicon.

A recognition lexicon should consist of probabilistic pronunciation networks that accurately model the variations in phonetic pronunciations observed in continuous speech. A pronunciation network of a particular sentence (1) should contain all allowable pronunciations of that sentence, (2) should not contain any pronunciations that are unreasonable, and (3) should contain probabilities that accurately reflect the true pronunciation probabilities.

The RULE system was designed to generate pronunciation networks by applying a set of lexical rules to a baseform network. In an earlier version of the RULE system (described in the paper presented at last year's DARPA meeting), when the phonological rules were applied to the baseform networks of a single word, the resulting network described the possible word-internal pronunciations of that single word. When the phonological rules were applied to the baseform networks of whole sentences, the resulting network described the variations in pronunciation of that whole sentence. Since the phonological rules were applied to a known sentence, particular word context effects were easily handled. We have extended the RULE system to apply a set of linguistic rules to individual word baseforms, and generate a pronunciation network of that word that represents all significant interword effects. We have also extended the RULE system to apply probabilistic phonological rules to a pronunciation network. The new algorithms keep track of the pronunciation probabilities and the identities of the rules that generated which pronunciation paths. We will describe these facilities, as well as the algorithms that can be used to automatically train the probabilities of a set of phonological rules.

In addition to the above extensions of RULE, other recent developments include: (1) a set of algorithms to convert a pronunciation network into a minimum deterministic network, and (2) an improved interactive graphical display to manipulate and inspect networks. These algorithms will not be described in this paper.

2. DEFINITIONS

A pronunciation network is a directed graph that RULE represents as a list of nodes and arcs. Each network has a single start-node and a single end-node. Each path through the network (from the start-node to the end-node) represents a possible pronunciation of a word or sentence. Since the pronunciation network does not contain loops, each network contains a finite number of different pronunciations.

The arcs of a network contain all the relevant information about the allowable pronunciations. Each arc contains the following information:

(1) Arc-Label: (e.g. "T" "D" "IY" "&" NULL-ARC) All the arc labels in the final version of a pronunciation network (after the rules have been applied) correspond to specific phonetic events. As we have implemented our rule set, secondary labels such as "&", "%", and computational constructs such as NULL-ARC can exist at intermediate stages of rule application, but are removed from the network by the application of lexical rules that delete these arcs.

(2) Arc-Features: (e.g. SYLLABIC CONSONANTAL STOP NASAL) These features represent the linguistic properties of this arc.

(3) Arc-Probabilities: These are used to represent the probability of different pronunciations. The sum of the probability of all arcs that leave each node should be equal to 1.0.

(4) Arc-Interword-Boundary-Constraints: Path constraints determine which of the pronunciation paths through the network are allowable. Whether or not a path is allowable depends on the preceding and following word context. Each arc that leaves from the start-node contains a LEFT-INTERWORD-BOUNDARY-CONSTRAINT. Each arc that arrives at the end-node contains a RIGHT-INTERWORD-BOUNDARY-CONSTRAINT. Other arcs in the network do not contain path constraints. When the right-interword-boundary-constraint of the last arc in one network is COMPATIBLE with the left-interword-boundary-constraint of the first arc in another network, the pronunciation paths are compatible, and are allowed to follow each other when parsing a sentence.

(5) Arc-Rule-Bookkeeping-Information: This information is used to keep track of which rules generated different pronunciation paths. This information allows the system to use hand-transcribed data to compute the probability of different rules.

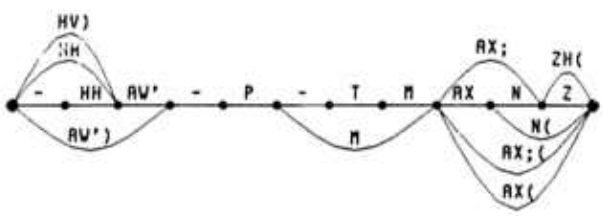


FIGURE 1. Sample Pronunciation Network of the word "HAUPTMAN'S"

[Note: ";" implies that this arc is nasalized, ")" implies: the previous word context determines whether this pronunciation path may or may not be taken, "(" implies: the following word context determines whether this pronunciation path may or may not be taken, AW' is AW with primary stress.]

We have constructed a dictionary of baseform representations for 3412 of the words in the TI-AP database. The baseform representation for each word is a network, and need not be a single string of symbols (i.e. it could be a multipath network). Each baseform network begins with an arc whose label ("&" or "%") represents a word boundary. Some sample baseform networks are shown in figure 2.

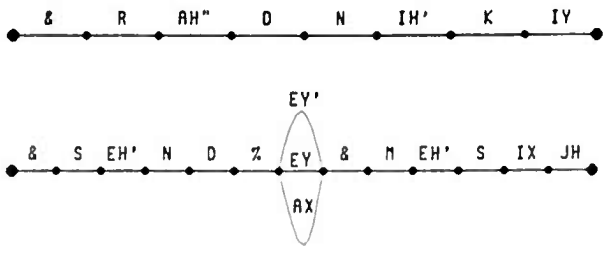


FIGURE 2. Baseform networks for (a) the isolated word "RUDNICKY", (b) the sentence "SEND A MESSAGE"

The interactive RULE facility allows a user to write a sequence of phonological rules. The current rule set consists of 45 rules. To illustrate the properties of phonological rules, an example is shown below. Rule V5 is automatically compiled (by the RULE system) into a list of rule clauses (figure 4). Each rule clause consists of: (1) a test that must be satisfied to match the clause to a single arc in the pronunciation network, (2) the action that is to be taken on the copied arc, and (3) a word boundary identifier which is used to determine where the rule can be broken up across word networks. The third clause of rule V5 is an optional morpheme-boundary test that was automatically inserted by the rule compiler. The insertion of the optional morpheme-boundary clause allows rule V5 to operate across word boundaries. Since this rule clause is optional, the rule application algorithm may or may not be match this clause to a network arc.

```
(defrule
   :name                          V5
   :rule-documentation            "W-GLIDE Vowel becomes
                                  SCHWA W"
   :core                          (feature W-GLIDE)
   :left-environment              NIL
   :right-environment             (feature-and SYLLABIC
                                  VOCALIC)
   :action                        ((replace-phoneme "AX")
                                  (insert "W"))
   :rule-type                     MIT
   :copy-matching-arcs            T
   :rule-probability              .5
   :application-order-number      2050
   )
```

FIGURE 3. A sample rule

TEST TO MATCH ARC	COPIED ARC ACTION	WORD BOUNDARY ID
1. (FEATURE W-GLIDE)	(REPLACE-PHONEME AX)	NONE
2. NONE	(INSERT W)	NONE
3. (OPTIONAL (FEATURE MORPHEME-BOUNDARY))	(DO-NOTHING)	V5-1
4. (FEATURE-AND SYLLABIC VOCALIC)	(DO-NOTHING)	NONE

FIGURE 4. The set of rule clauses that rule V5 is compiled into.

The word boundary identifiers are used (1) to indicate to the rule application algorithm where the rule clauses can be split across individual word networks, and (2) in the interword boundary constraints that determine which pronunciation paths of one network can follow the pronunciation paths of the previous network. All rule clauses that can match the initial word boundary symbol of a pronunciation network are "possible break points" where a phonological rule can be split across networks. Since each baseform pronunciation network starts with a word boundary symbol (either an "&" or an "%"), those rule clauses that can match these arcs are given a UNIQUE word boundary identifier. In rule V5, the third clause is given the unique interword boundary identifier V5-1. Each rule can be split across pronunciation networks at the location between the clause with an interword boundary identifier, and the previous clause. In rule V5, this is between the second and third clauses.

Each initial arc and each final arc of a word pronunciation network contain an interword boundary constraint. These interword boundary constraints determine whether a pronunciation path that begins one individual word network is allowed to follow a pronunciation path that ends another word network. The pronunciation paths of the two word networks can follow each other if the two interword boundary constraints (of the last arc in network-1, and the first arc in network-2) are COMPATIBLE. Each boundary constraint consists of a list of single boundary constraints. Each single boundary constraints contain two lists: a list of OPTIONAL word boundary identifiers, and a list of OBLIGATORY word boundary identifiers. For two boundary constraints to be compatible, each obligatory word boundary identifier in one interword constraint must be contained in the list of either the obligatory or the optional word boundary identifier of the other interword constraint.

3. PROBABILITIES

The pronunciation network of a word initially consists of the baseform network. Each of the phonological rules are applied sequentially to each baseform network. To apply a rule to the network, the application algorithm searches the network for a series of arcs that satisfy the list of rule clauses. When a series of arcs are found that match the rule clauses, the matching arcs are copied, and the (1) labels, (2) features, (3) interword boundary constraints, and (4) path probability of these copied arcs are modified. The path probability of the matching arc sequence is multiplied by the probability of the rule. The new arcs represent an alternate pronunciation path. Since the rules are applied sequentially, the pronunciation paths generated by previous rules may be used to match the rule clauses of following phonological rules.

To illustrate how a rule is applied to a network, we can look at figures 5 and 6. The purpose of sample rule RULE-1 is to allow the alternate pronunciation "E" to a series of arcs ("A" "C"). The 4 stages of rule application are illustrated in figure 6: (1) the network is exhaustively searched for all series of arcs that match the rule; for each of those series of arcs that match, the following steps are taken: (2) the matching arcs of the network are broken out into a separate linear path, (3) the matching arcs are copied and subsequently modified by the actions of the rule clauses, (4) the network is converted into a minimum deterministic graph. The algorithm is described in much greater detail in Appendix 1.

```
(defrule
   :name                        RULE-1
   :rule-documentation          "illustrative example"
   :core                        ((phoneme 'A") (phoneme "C"))
   :left-environment            NIL
   :right-environment           NIL
   :action                      ((replace-phoneme "E")
                                (delete-phoneme))
   :rule-type                   TEST
   :copy-matching-arcs          T
   :rule-probability            .5
   :application-order-number    1
   )
```

FIGURE 5. A sample rule that will be used to demonstrate how rules are applied to the pronunciation network.

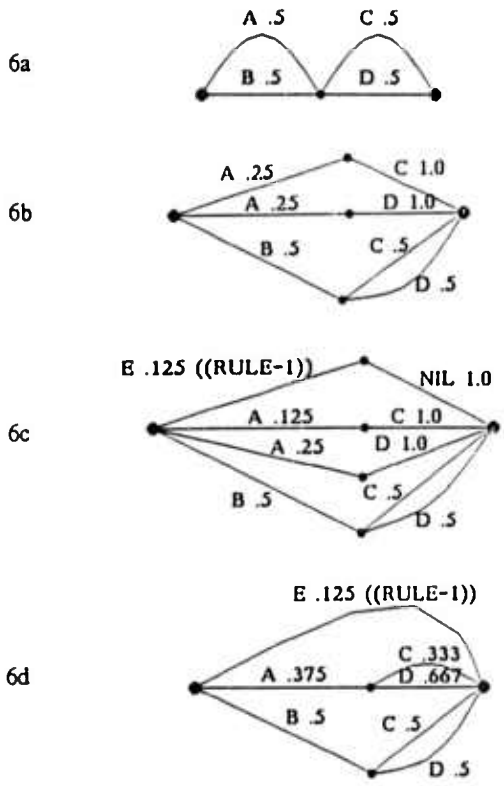


FIGURE 6. The sample rule in figure 5 is applied to a network.

6a: The original network.
6b: The clauses in RULE-1 have been matched against the network, and the matching path ("A" "C") has been expanded into a linear path.
6c: The matching path ("A" "C") is copied, and the appropriate actions are taken on these arcs. The network probabilities are modified, and the rule bookkeeping indicates which rule generated this new path.
6d: The network is converted into a minimum deterministic graph.

After the rule clauses have been successfully matched to a sequence of arcs in the network, the network is expanded into a series of linear arc sequences (see figure 6, top right). This network expansion is necessary so that the pronunciation probabilities can be modified in a correctly. The original matched arcs are then copied, and modified by the actions specified in the rule. The path probability of the newly modified arc sequence is equal to the probability of the original matched arc sequence multiplied by the rule probability. The path probability of the original matched arc sequence is multiplied by (- 1.0 rule probability). Finally, the network is converted into a minimum deterministic graph, maintaining the correct path probabilities and rule bookkeeping information. This algorithm is described in more detail in appendix 1.

To compute the probability of each phonological rule, a database of hand transcribed speech is necessary. For each utterance in the database, a set of phonological rules is applied to a sentence baseform network to create a pronunciation network for that sentence. Using the hand transcribed data, the pronunciation path through the network is computed. Beginning at the

start-node, RULE traverses the pronunciation network along the observed path. At each node along this path, RULE can compute a list of all the different phonological rules that generated any of the arcs that leave from the node. For each of the rules that are in this list, we increment their POSSIBLY-APPLIED-COUNT by 1. We then increment the ACTUALLY-APPLIED-COUNT of all the phonological rules that were used to generate the traversed arc. When we have finished processing all the utterances in the database, the probability of each phonological rule is equ. . (// ACTUALLY-APPLIED-COUNT POSSIBLY-APPLIED-COUNT).

4. INTERWORD RULE APPLICATION

To allow phonological rules to apply across words when each word is stored in a lexicon as a separate virtual network, two major changes were made. These changes consisted of (1) a new rule application strategy, and (2) the addition of an interword boundary constraint that determined which pronunciations of a word were possible, based on the previous/following word context.

Each rule consists of a series of rule clauses. During rule application, each rule clause needs to be matched against an arc in the network. In order to deal with interword effects, the new rule application algorithm needs to allow partial sequences of rule clauses to be matched against a series of arcs in the network, with the remaining clauses to be applied to the previous/following word. Since each baseform pronunciation network of an individual word starts with a word boundary symbol (either an "&" or an "%"), only rule clauses that can match these arcs can split a rule across a network boundary. Therefore, each rule clause that can match these arcs is given a unique word boundary identifier. Rule clauses that cannot match a word boundary symbol, cannot be locations where a rule is broken up across word boundaries. Each rule can be split across pronunciation networks at the location between the clause with an interword boundary identifier, and the previous clause. In rule V5, this is between the second and third clauses. An example of how rule V5 (see figure 3) can be broken up across word boundaries is shown in figure 7.

TEST TO MATCH ARC	COPIED ARC ACTION	WORD BOUNDARY ID
1. (FEATURE W-GLIDE)	(REPLACE-PHONEME AX)	NONE
2. NONE	(INSERT W)	NONE
*** BREAK ACROSS NETWORKS HERE ***		
3. (OPTIONAL (FEATURE MORPHEME-BOUNDARY))	(DO-NOTHING)	V5-1
4. (FEATURE-AND SYLLABIC VOCALIC)	(DO-NOTHING)	NONE

FIGURE 7. How the clauses of rule V5 can be split across word boundaries. To apply this rule across network boundaries, the first clause matches the last arc in network-1, while the third clause matches the first arc in network-2. Since the second clause is an insertion, it does not need to match anything in either network. The changes in pronunciation (both the AX and the W) will be associated with network-1.

The rule application algorithm was modified to allow partial sequences of rule clauses to be applied to networks. When the rule application algorithm (in appendix 1) reaches the end of a network, and also encounters an interword boundary identifier on the next clause to match the network, it allows the rule to be split across networks. An example of rule V5 being split across network boundaries is shown in figure 8.

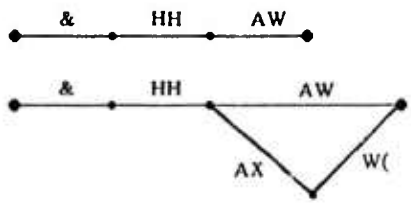


FIGURE 8. Top: The baseform network for the word "HOW". Bottom: After rule V5 has been applied to the network. The new pronunciation path that has been added is highlighted. The "(" of the "W(" means that this pronunciation path can only be taken if the following word satisfies certain conditions, in this case that the beginning of the following word network has a syllabic vocalic arc just inside the word boundary as specified by the last two clauses of rule V5.

The new pronunciation path ["AX", "W"] of the word "HOW" in figure 8, may only be traversed if the following word satisfies the interword boundary constraint. This is because the new pronunciation path is conditional on the features of the word that follows it. The new pronunciation of the word "HOW" may only be used if the next pronunciation network starts with a morpheme boundary followed by an arc that is both syllabic and vocalic. This interword boundary constraint indicates where the rules were split across network boundaries, and which word-edge paths are consistent with each other.

5. BASEFORMS AND PHONOLOGICAL RULES IMPLEMENTED IN RULE

For CMU's Electronic Mail Task, SRI implemented a recognition lexicon that recognizes multiple pronunciations of most vocabulary words. The variant pronunciations (e.g. "decision" with or without a tense first vowel, or "capacity" with flap or an aspirated [t]) can be directly represented in the baseform list or they can be derived by rule from single baseforms. In SRI's work thus far, we have maintained an intuitive, but principled split between irregular or lexical variants, and general regular, rule governed variation. By this criterion, the forms of "exit" with voiced [gz] or voiceless [ks] are explicit in the baseform list; while the flap and [t] forms of "capacity" are handled by rule. This kind of split is possible in RULE but not required.

As of this writing, the rule set that SRI has implemented consists of about 45 rules that are separated into eight groups. The rules within a group apply more or less in parallel, while the members of lower numbered groups apply before members of higher numbered groups. The groups are:

0: Expansions -- Convenient redundancies like insertion of silences and glottal stops as appropriate at word boundaries.

1: Lexical and Dialectal Variants -- Regular dialectal and free variant forms such as /w/ or /wh/ in "where" etc., or initial-syllable tense-lax alternations in words like "demand" and "deny".

2: Syllabic Nucleus Core -- Deletion of unstressed initial vowels and the re-coding of diphthongs into schwa-glide sequences.

3 H and Glide or Liquid Core -- Deletions of /h/ and /l/ in certain environments.

4: Nasal Cores -- Assimilation of certain nasals.

5: Fricative Core -- Epenthetic stop insertion, and several assimilations.

6: Plosive Core -- Several kinds of lenition, including deletions, assimilations and cluster reduction.

7: Phonetic Alternates and Patches -- A catchall for covering regular (non-phonological) correspondences created by the logic of the Acoustic-Phonetics module, and for the deletion of diacritics such as boundary marks.

A final note: There are several types of phonological rules that are often used in linguistic descriptions and would be convenient for building a recognition lexicon that are not easily implemented in the RULE system. One example is "alpha" rule notation in which a variable, alpha, is bound in one clause of the rule and referred to in another. Alpha rules are useful in handling assimilations and geminate reductions. A last example is the use of abstract phones in cross dialectal baseforms; that is phones that are not realized in either dialect but are convenient ways to represent a regular correspondence between forms. Abstract phones and alpha rules can be done in RULE but they are not natural to its formalism.

6. SUMMARY

The RULE software system is a set of tools that allows one to construct recognition lexicons. This paper describes new algorithms that apply phonological rules to pronunciation networks. The novel aspects of the algorithm involve: (1) rule application to probabilistic pronunciation networks, and (2) generation of interword phonological effects when phonological rules are applied to individual word models. With a hand transcribed database, we can automatically train the probability of each of the phonological rules, and use these phonological rules to create accurate probabilistic pronunciation networks for each word in the vocabulary.

APPENDIX 1: ALGORITHM FOR APPLYING A SET OF RULES TO THE NETWORK

```
FUNCTION: APPLY-SET-OF-RULES-TO-NETWORK
      (list-of-ordered-rules,network)

Get the first rule of the list-of-ordered-rules

     Determine rules to apply in parallel with this rule =>
     PARALLEL-RULE-LIST
     REMAINING-RULE-LIST contains rules that remain.

RULE-APPLICATION-PASS # 1:
     Loop for each rule in parallel-rule-list
          Loop for each node in the network.
               If this is the start-node of the network,
                    THEN
                         Loop for each rule clause that has a word
                         boundary identifier
                              CALL: MATCH-CLAUSE-LIST-TO-
                              NETWORK
                                   (remaining-rule-clause-list,node)
                    ELSE
                              CALL: MATCH-CLAUSE-LIST-TO-
                              NETWORK (rule-clause-list,node)

Collect the arc sequences successfully matched by rule clauses. Expand
network so that all matched arc sequences are separated out.

RULE-APPLICATION-PASS # 2: [same loops and calls as pass #2]

Convert the network into a minimum deterministic graph.

If there are any rules remaining,
     THEN CALL: APPLY-SET-OF-RULES-TO-NETWORK
     (remaining-rule-list,network)


FUNCTION: MATCH-CLAUSE-LIST-TO-NETWORK
     (rule-clause-list,node)

If (OR [ rule-clause-list is empty ]
     (AND     [ node is the end-node of the network ]
              [ the next rule clause that has an arc test
                   also has a word boundary identifier]
     ))
     THEN
          CALL:MODIFY-NETWORK-BY-APPLYING-RULE
     ELSE
          Loop for arcs that leave this node
          If arc satisfies first remaining rule-clause,
            THEN
                 CALL: MODIFY-NETWORK-BY-
                 APPLYING-RULE
                 ((cdr rule-clause-list),(to-node arc))
            ELSE
                 If rule-clause is an optional clause,
                      THEN
                           CALL: MODIFY-NETWORK-
                           BY-APPLYING-RULE
                           ((cdr rule-clause-list),node)


FUNCTION: MODIFY-NETWORK-BY-APPLYING-RULE
          (network,rule-application-pass-number,
          unmatched-left-rule-clauses,matched-rule-clauses,
          unmatched-right-rule-clauses)

If first pass, then collect matched arcs into a temp data structure. If this is
the second pass of the rule application,
     THEN

          If there are no unmatched left or right rule clauses,
               (rule does not apply across network boundaries.)
               THEN
                    If the rule copies the matching arcs,
                    then copy the arcs, modify the probability and
                    bookkeeping of this path.
          Apply action to each copied arc (typically modify
               arc label or features).
     ELSE
          (there are some unmatched rule clauses,
           rule that applies across network boundaries.)

          Loop through matched rule clauses.
          If clause action modifies the network,
            THEN
                 If the rule copies the matching arcs, then copy the
                      arcs, modify the probability and bookkeeping of
                      this path.
                 Apply action to each copied arc (typically
                      modify arc label or features).
                 Modify interword constraints of copied arcs to contain
                      obligatory interword identifier.
            ELSE
                 Modify interword constraints of the matched arcs to
                      contain optional interword identifier.
```

STUDIES FOR AN ADAPTIVE RECOGNITION LEXICON[1]

Michael Cohen, Gay Baldwin, Jared Bernstein, Hy Murveit, Mitchel Weintraub
Speech Research Program
SRI International
Menlo Park, CA 94025

ABSTRACT

In the past year, SRI has undertaken a series of empirical studies of phonological variation. The goal has been to find better lexical representations of the structure and variation of real speech, in order to provide speaker independence in speech recognition. Results from these studies indicate that knowledge of probabilities of occurrence of allophonic forms, co-occurrence of allophonic forms, and speaker pronunciation groups can be used to lower lexical entropy (i.e., improve predictive ability of lexical models), and possibly, therefore, achieve rapid initial adaptation to a new speaker as well as ongoing adaptation to a single speaker.

INTRODUCTION

As the number of words in the lexicon grows, the speech recognition problem gets more difficult. In a similar way, as more possible pronunciations for each word are included in the lexicon, the recognition problem gets more difficult because there are more competing hypotheses and there can be more overlap between the representations of similar words.

One important goal of a lexical representation is to maximize coverage of the pronunciations the system will have to deal with, while minimizing overcoverage. Overcoverage adds unnecessary difficulty to the recognition problem. One way to maximize coverage while minimizing overcoverage is to explicitly represent all possible pronunciations of each vocabulary word as a network of allophones. An example of such a network is shown in figure 1 for the word "water". This network represents eight possible pronunciations, some of which are fairly common (e.g., [W AO DX ER]), and others somewhat rare (e.g., [W AA T AX]). Experience suggests that, to assure coverage, it will be necessary to include many pronunciations for each word, including those which happen relatively rarely.

In reality, speech is more highly organized. There is more predictive knowledge available than in a model that simply represents independent equiprobable choices with no interaction or influence between different parts of a model and with no ability to use information from other parts of an utterance or previous utterances by the current speaker. In current systems which use allophonic models, each node represents an independent set of equiprobable choices.

[1]This research was sponsored by Defense Advanced Research Projects Agency Contract N00039-85-C-0302. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

The goal of the research described in this paper is to explore ways in which a lexical representation can better reflect the structure of real speech data, so that the representation will have more predictive power, and thus improve recognition accuracy. A better understanding of the issues involved may lead to methods for rapid adaptation to a new speaker, as well as ongoing adaptation to a single speaker during a single session.

In order to explore these issues, we chose to model (as a single utterance) a pair of sentences containing 21 words for which we had a large data set. The patterns of variation found for this 21-word microcosm should indicate what kinds of structures will he needed in a larger lexicon. The data used were transcriptions of the two dialect sentences for the 630 speakers in the TI-AP database.

We have performed a series of four studies that explored four types of phonological structure, and ways of representing this structure in a lexicon. In the first study, we simply looked at the gain in predictive ability of a phonological model which incorporates knowledge of the probabilities of the various possible word pronunciations. The second study explored the co-occurrence of allophonic forms, and ways in which knowledge of these co-occurrences can be automatically compiled into a phonological model. The third study explored the possibility of grouping speakers into a small number of pronunciation clusters, and looked for demographic and other predictors of these pronunciation clusters. The fourth study was designed to compare intra-speaker variation to the variation within the pronunciation clusters defined hy the third study.

To evaluate our data, and compare representations, we used entropy as a measure of the predictive power of a representation, or difficulty of the recognition task given a particular representation. The entropy of a representation, developed from or "trained" on some large set of data, reflects both how well the representation captures significant structure in the data and how much predictive power is gained by modelling this structure.

The four studies are described in the following four sections, followed by a general discussion and conclusions.

PRONUNCIATION PROBABILITIES

The goal of the first study was to determine how much speech recognition accuracy could be improved by incorporating knowledge of pronunciation probabilities into a phonological language model. An important goal of any lexical representation is to provide coverage of the pronunciations that the system will have to to deal with, including relatively rare pronunciations. This makes the recognition problem more difficult because there are more competing hypotheses and can be more overlap between word models. One way to deal with this problem is to include probabilities for pronunciations in the lexical model. In this way, including somewhat rare pronunciations will increase coverage without hurting performance. It will

allow recognition of these unusual pronunciations, avoiding confusion with other more common pronunciations of similar words. For example, consider the allophone string

[DH AX B IH G W AA DX AX B IH L Z]

This string contains, as a substring, the sequence [W AA DX AX], which corresponds to one of the paths through the network for the word "water" shown in figure 1. This is a relatively infrequent pronunciation of the word "water". An alternative hypothesis for this same substring could be the pair of words "wad of", for which this pronunciation is relatively common. (Suggesting the phrase "The big wad of bills" rather than "The big water bills".) Appropriate probabilities associated with these pronunciations could allow a system to make a more intelligent choice. Such a model should help recognition accuracy significantly, provided that the probabilities used are accurate for the domain in which the system will be used, and especially if the probability distributions are significantly different from the default equi-probable models.

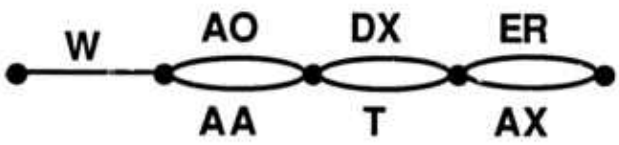
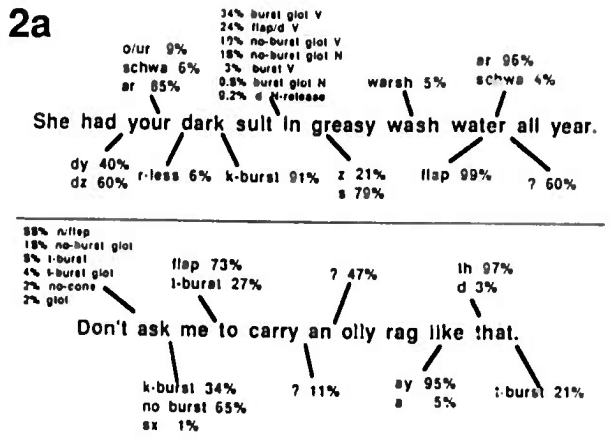


Figure 1. Allophone network for the word "water".

The data used in this study were transcriptions of the two dialect sentences for the 630 speakers in the TI-AP database. Originally, the allophonic forms used for each of 58 phonemes in the two test sentences as produced hy the 630 speakers were transcribed by Margaret Kahn, Jared Bernstein, or Gay Baldwin. The transcriptions were done carefully using a high fidelity interactive waveform editor with a convenient means to mark and play regions in a high resolution image of the waveform. Spectrographic and other analytic displays were also easily available, though most of the work was done by ear and by visual inspection of the waveforms. Subsequently, a subset of 18 of these seginents was chosen which we felt we could transcribe accurately and consistently; the 18 are disproportionately consonantal. This subset of 18 segments in the original 630 transcriptions were then re-checked and corrected by one individual (Gay Baldwin). The transcriptions of these 18 segments/utterance were compared to a subset of 156 speakers whose sentences had heen independently transcribed at MIT as part of a related project. For this subset of 156 speakers, the number of transcription disagreements between SRI and MIT wa, about 5-10% for a typical phoneme.

Figure 2a shows the two dialect sentences, indicating the segments included in this study, along with the distributions of allophones found for each of these phonemes. Figure 2b shows the 18 node allophone model used to represent the possible pronunciations. Among these 18 phonemes, at the level of transcription we used, there are 14 two-way splits, two three-way splits, and a six and a seven-way split. The distributions vary from a 1%-99% split for canonical /t/ vs. flap in "water" to a 60%-40% split for the affricated vs. non-affricated /dy/ juncture in "had-your" to a 47%-53% split for a glottal gesture at the beginning of "oily".

The seven-way split for the juncture in "suit in" is the most unpredictable. The potentially variable events are the burst of the /t/, the occurrence of a glottal onset to "in" and the presence or ahsence of the vowel in "in". A third of the readers produced a very



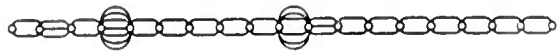


Figure 2.
a) observed percentages of allophonic forms.
b) 18 node utterance model.

clear form that exhibited a t-burst and a glottal stop or glottalization at the onset of the /I/ in "in". A quarter of the readers flapped or produced a short /d/ into the vowel in "in". Nineteen percent of the utterances showed no burst for the /t/ but a glottal gesture into the /I/, while 18% showed the same burstless /t/ with glottal gesture, but released the gesture directly into the nasal, deleting the /I/. Three percent of the readers (19 people) released the /t/ with a burst right into the /I/, 3 speakers (0.6%) had a clear t-burst, but the glottal gesture goes right into the nasal with the /I/ deleted. One speaker (0.2% of the sample) produced the "suit in" juncture as a /d/ with a velic release into the nasal (as in a word like "sudden"). The distributions are surprising only as reminders of how little quantitative data on the relative frequency of occurrence of allophones is available. What experienced phonetician could have estimated the proportion of these forms in reading? It's no wonder that speech recognition lexicons would have whatever allophonic options they allow unspecified as to relative likelihood.

The approach used in this study was to compute the probabilities of each of the transitions in the 18-node allophone model (figure 2b) from a large database of speech. The entropy of this model was computed, and compared to the entropy of a similar model without probabilities estimated from data, in which case all transitions from a node are considered equiprobahle. Information theoretic entropy, H, of an arbitrary string, S, in the language was computed as:

$$H(S) = -\sum_{n}\sum_{t} P(t)\log_2 P(t) \qquad (1)$$

where n ranged over all of the the nodes in the utterance model, t ranged over all of the transitions from the current node, and P(t) is the probability of transition t. This is the same as:

$$H(S) = -\sum_{s} P(s)\log_2 P(s) \qquad (2)$$

where s ranges over all of the strings in the language [McEliece, 1977].

The entropy measured for the model with equi-probable transitions is 22.6 bits, and for the model with empirically estimated probabilities is 13.0 bits. This represents an increase of 42.4% in predictive ability (knowledge or source of constraint) for the model with trained probabilities. Presumably, this further constraint should translate into improved recognition accuracy.

CO-OCCURRENCE OF ALLOPHONIC FORMS

The goal of the next study was to explore co-occurrence relationships in allophonic variation. A co-occurrence relationship is one in which the probability of the occurrence of a particular variant is conditioned on the presence or absence of some other variant in another part of the utterance. Knowledge of such co-occurrence relationships can be used to increase predictive power about allophonic variation.

The data used in this study was the same as that used in the previous study, except that the realization of /k/ in "dark" was excluded, since we decided we had insufficient confidence in our transcriptions of that phoneme. All possible pairings of the remaining 17 phonemes (136 pairs) were tested for co-occurrence relationships. The two examples in figure 3 demonstrate the technique. For each pair of segments, counts of all combinations of variants for the two forms were entered into a matrix. Chi-square tests were performed on these matrices at the 97.5% confidence level.

The example in figure 3a illustrates the analysis of glottal (or no glottal) at the beginning of "all" and "oily". The table shows that, of the 630 speakers, 230 used a glottal gesture (either a full glottal stop or a weaker gesture seen as several irregular glottal periods, both symbolized here as [ʔ]) at the beginning of both "all" and "oily". One hundred eighty-four speakers didn't use [ʔ] before either word, 65 speakers put [ʔ] just on "oily", and 151 just on "all". The chi-square is significant at the 97.5% level, indicating that this pattern of co-occurrence of glottals at the beginning of "all" and "oily" is rather unlikely to happen by chance if we assume that the two events are independent. In other words, speakers who used [ʔ] before "all" were more likely to use [ʔ] before "oily" as well. Similarly, if [ʔ] was omitted before "all", it was less likely to be found before "oily". This case of co-occurrence is not surprising, because both forms could be considered to result from the same phonological rule.

The co-occurrence matrix in figure 3b shows a dependent relationship between forms that are phonologically heterogeneous. In this case speakers who use [t] rather than flap in "to" show a strong tendency to use (rather than omit) [ʔ] before "an". This might be interpreted as evidence for a higher level fast-speech (or lax style) "macro-rule", which increases the likelihood of several types of phonological rules. One goal of our work is to establish a method by which such functional rule groups can be found (or dismissed). For now we just present preliminary data that show non-independence between pairs of forms over this sample of utterances.

Figure 4 shows which of the 136 possible co-occurrences actually had chi-squared values that indicated non-independence. The confidence level for the chi-squared value was 97.5%, meaning that of the 136 chi-squares calculated, one could expect about four artifactually non-independence between co-occurring forms. Of these 37, about 15 involve pairs that have a clear phonological relation, (r-lessness in "your", "dark", "water"; [ʔ] in "all", "an", "oily"; flapping in "water", "to", "don't ask", "suit in"; etc.). Most of the remainder show dependencies between variants in more remotely related phonological contexts. The number of dependencies is obviously considerable, and suggests that macro-level relationships – dialect region, utterance speed, style, sex-linked variation – are pervasive enough to be useful in improving predictions of forms for automatic speech recognition.

3a

oily-? \ all-?	0	?
?	65	230
0	184	151

3b

to-t \ an-?	?	0
burst	39	129
flap/d	28	434

Figure 3. Co-occurrence Examples: a) onsets in "all" and "oily". b) onsets for "an" and "to".

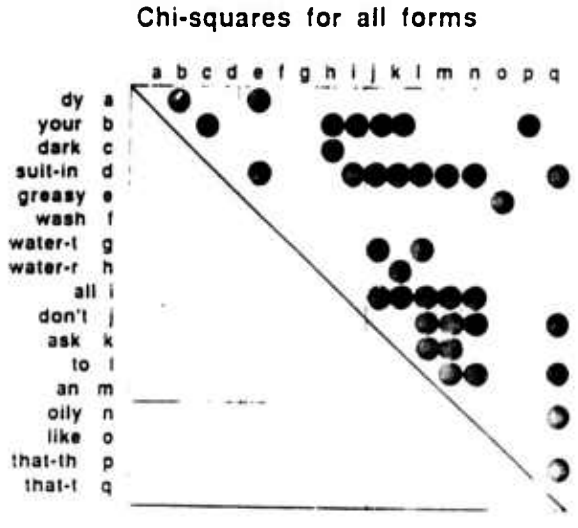


Figure 4. Significant Chi-squares for form pairs.

There could be significant advantage in finding a way to compile knowledge about the co-occurrence of allophonic forms into the phonological model. This would allow a form of within-utterance adaptation to take place automatically. An example of how this might be done is shown in figure 5. Figure 5a shows a probabilistic language model for a language consisting of strings of two symbols, the first symbol being "A" half the time and "B" the other half of the time, and the second symbol evenly divided between "C" and "D". There is additional structure to this language, in the form of co-occurrence. When the first symbol is "A", the probability is 90% that the second symbol will be "C", and 10% that it will be "D". When the first symbol is "B", the distribution is reversed. The entropy of such a model can be calculated as 1.47 bits. When a model representing the same language is configured as in figure 5b, without representation of the co-occurrence, the entropy is two bits, one bit for the choice at each node. Configuring the model to reflect co-occurrence knowledge has resulted in more than 25% lower entropy. Clearly, the model in 5a can do a better job of predicting incoming strings in the language than that in 5b.

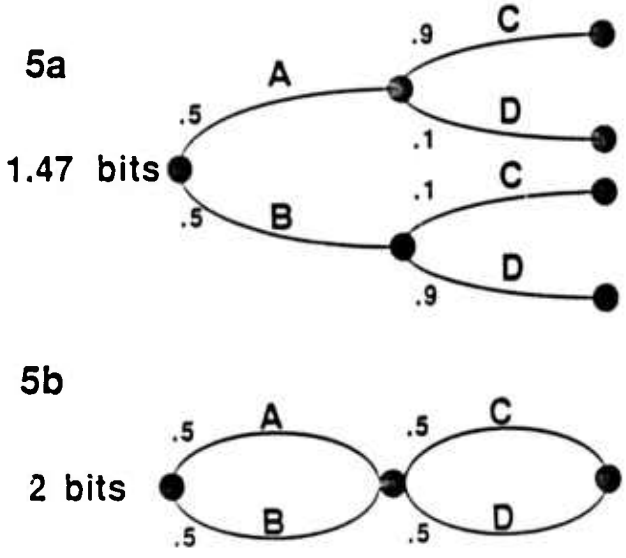


Figure 5. Alternative models for a simple language.

We performed a clustering study to determine whether we could compile co-occurrence knowledge into a phonological model, hence lowering entropy, using an automatic procedure. Considering each sentence pair from a speaker to constitute one utterance, we clustered the 630 utterances into the lowest entropy groups we could find. Each group could then be used to estimate allophone probabilities for an independent path through the model (see figure 6). Grouping together utterances with similar allophonic realizations in this manner allows the phonological model to capture co-occurrence knowledge by isolating co-occurring allophones in the same paths. If there is significant co-occurrence in the data, this new model should have lower entropy, and hence greater predictive power, than the previous model.

The clustering technique used was a combination of hierarchical clustering and the iterative Lloyd algorithm [Duda and Hart, 1973]. For each specific number of clusters desired, the data were clustered into that number of groups using an agglomerative hierarchical clustering technique, and then these clusters were used as the seeds to

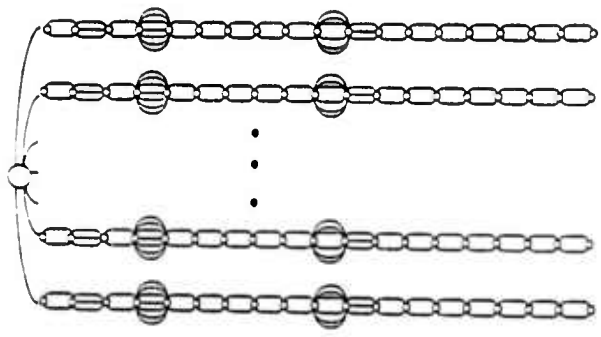


Figure 6. Allophone network for the 2-sentence utterance, showing clusters as separate paths.

the Lloyd algorithm. Each step of the hierarchical clustering algorithm involves merging the nearest pair of distinct clusters. Initially, each utterance forms a singleton cluster, and the procedure continues until the desired number of clusters is reached. At each step, the nearest pair of clusters was defined as that pair whose merging would result in a model with the lowest conditional entropy H(S|c), which was computed as:

$$H(S \mid c) = \frac{1}{N}\sum_{i=1}^{n} M(i)H(S \mid i) \qquad (3)$$

where N = *total number of utterances in the sample* (630),

n = *current number of clusters*,

$M(i)$ = *number of utterances in cluster* i, and

$H(S \mid i)$ = *entropy of a string* S *in cluster* i.

Hence, H(S|c) is defined as the weighted average (weighted by cluster size) of the entropies of the individual clusters, which is the same as the entropy of a string, given that you know which cluster the string falls into. Though the real objective of this procedure was to minimize H(S) rather than H(S|c), computing H(S) for the composite model at each iteration of the algorithm is computationally too expensive. Though H(S|c) is not guaranteed to be monotonically related to H(S), it should be in most cases.

In the second phase of clustering, the clusters found by hierarchical clustering were used as a seed to the iterative Lloyd algorithm, which continued until the improvement for one iteration was less than a threshold. Each iteration of the Lloyd algorithm involved the following:

1) For each utterance: compute H(S|c), as in equation 3, with this utterance as a member of each current cluster - remember the cluster for which H(S|c) is minimal.

2) Once the new cluster is chosen for all utterances, actually make the switches.

Typically, the Lloyd algorithm continued for 5-10 iterations, and the amount of reduction in H(S) over the clusters output from the hierarchical clustering procedure was another 1-2% lower than the unclustered model.

The results of the clustering study are shown in figure 7. The higher curve H(S) is for the composite model given 10, 20, and 30 clusters. The results show that the entropy of a phonological model can be lowered 10-15% by modelling the co-occurrence of allophonic forms. Furthermore, this co-occurrence can be modelled for any sufficiently large data set by running a standard clustering algorithm, without the need to explicitly determine what the co-occurrences are. The significance of the lower curve (H(S|c)) will be discussed below in the section on speaker groups.

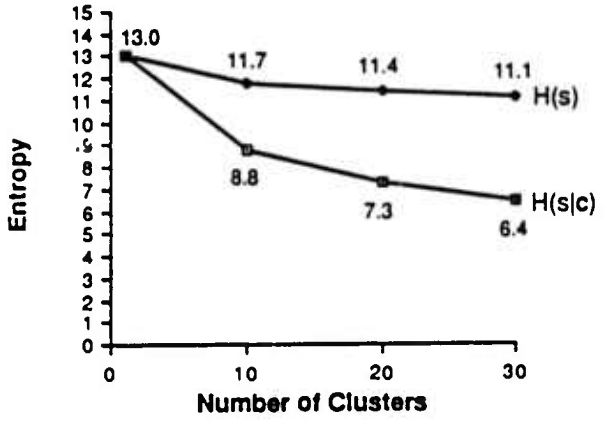


Figure 7. Entropy of utterance model as a function of the number of clusters.

We also tested whether demographic factors and speech rate could be used to predict allophonic forms. These results are shown in figure 8. Chi-squares (at the 97.5% confidence level) were computed to test for independence between region (each speaker was identified with one of seven geographic regions or as an "army brat"), age (by decade), race, sex, education (HS, BS, MS, or PhD), and speech rate, vs. form. As can be seen, the results show significant non-independence between all of the demographic factors vs. form and rate vs. form. This indicates that all of these factors are significant predictors of allophonic occurrences. For example, people from New England tend to say r-less "your", and people from the South tend to say "greasy" with a [z].

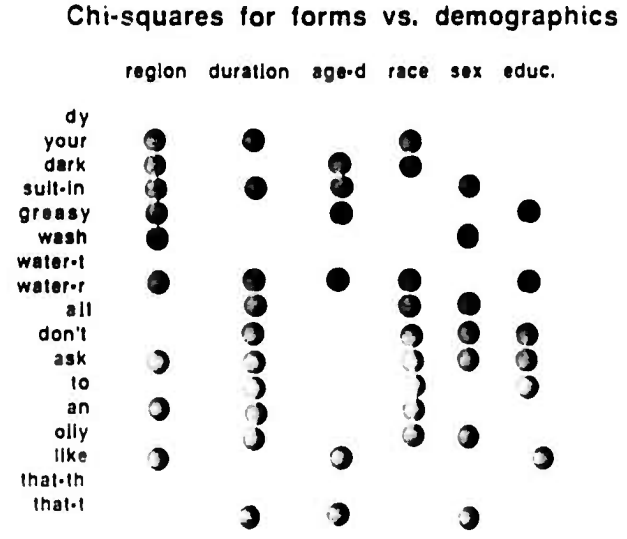


Figure 8.

SPEAKER GROUPS

The lower curve in figure 7 shows the conditional entropy of the model given the cluster, computed as in equation 3. This result indicates that if the appropriate cluster for the incoming utterance is known in advance, entropy can be lowered 30-50% from the unclustered model. The question then arises of how well can we predict the cluster for an incoming utterance. This question, in turn, raises a number of additional questions:

1) What explicit predictors of cluster membership are available? (e.g., sex, region of origin, speech rate, etc.)

2) How consistently does a speaker stay within one cluster? (i.e., If a speaker stays in the same cluster with reasonable consistency, then rapid adaptation to a new speaker may be accomplished by choosing the appropriate cluster after some experience with this speaker, or choosing an appropriate weighting function over the clusters.)

3) How can we classify a speaker into the appropriate cluster, or choose the appropriate weighting function over clusters for this speaker at the current time?

4) When during a recognition session should a new cluster be chosen, or a new weighting function be computed? (e.g., when speech rate changes, when performance drops, only when a new speaker comes along, etc.)

The studies described in this section were designed to address the first two questions.

In order to test for predictors of cluster membership, we performed chi-squares at the 97.5% confidence level, testing for non-independence between cluster vs. form, cluster vs. all of our demographic factors (age, race, region, sex, and education), and cluster vs. speech rate. There was significant non-independence between cluster and all allophonic forms except for the /t/ in "water", as well as for all demographic factors and rate. The lack of significance for /t/ in "water" is not surprising since, out of our sample of 630 speakers, only five of them aspirated the /t/.

In order to test the consistency with which speakers remain in clusters, we gathered a new set of data, consisting of speakers repeating the same sentences many times. Four speakers were recorded in three sessions each, with recording sessions for the same speaker a week apart. The recordings were made in a sound-treated room, using a close talking microphone and a Nagra tape recorder. Each recording session consisted of eight readings of the same two sentences used in the experiments described earlier, interspersed in a set of seven filler sentences. The first five repetitions were uninstructed (i.e., "normal reading"). At the sixth repetition, the subjects were instructed to read very quickly, at the seventh slowly and carefully, and at the eighth normally. From listening to the recordings, it is our judgement that the fast readings were, indeed, extremely fast, and the slow and careful readings were extremely slow and careful. Since the uninstructed readings were fairly fast, the differences between the slow and uninstructed readings were more dramatic than those between the fast and uninstructed readings. The final data set consists of 96 repetitions of the two sentences, 24 from each speaker, with 72 repetitions uninstructed or "normal", 12 fast, and 12 slow and careful.

The same 18 phonemes used in the earlier experiments were phonetically transcribed, with the aid of the tools described earlier, by Michael Cohen, and checked by Jared Bernstein and Gay Baldwin. Each of the 96 utterances were then classified into the clusters based on the 630-speaker data, as described in the previous section. We chose to classify them into the 10-cluster version so that each cluster would be based on a large number of utterances (approximately 63). Each utterance was classified into the cluster with the centroid with

minimal Euclidean distance to the utterance. Table 1 shows the number of utterances for each speaker classified into each cluster. As can he seen, most of the utterances for each speaker tend to be classified into two or three clusters. Eleven of the 12 slow utterances were classified into cluster two. The fast utterances did not tend to fall into any one cluster.

Table 1. Classification of speaker utterances according to pre-existing clusters

Speakers	Clusters									
	0	1	2	3	4	5	6	7	8	9
JB	0	0	3	0	1	2	3	1	0	14
JK	0	2	6	1	5	0	0	1	9	0
KC	0	1	2	1	2	0	1	0	0	17
PR	0	11	8	0	0	0	3	2	0	0

These results indicate that although speakers fall into the same clusters with some consistency, choosing a single cluster for a speaker is inadequate. A more reasonable approach may be to choose a weighting function over all of the clusters. Furthermore, cluster membership seems to be somewhat dependent on speech rate.

INTRA-SPEAKER VS. INTRA-GROUP VARIATION

The results of the previous section suggest a method of adaptation by choosing appropriate sets of (or weights for) clusters for a speaker. The study described in this section addresses the question of whether or not it is useful to try to further adapt to the individual speaker once the clusters are chosen. We have addressed that question by comparing the amount of variation within a single speaker to the amount of variation within a single cluster. If there is considerably less variation within a speaker than within even a single cluster, then there may be ways to further adapt to the individual speaker.

The data used for this experiment included both the 630-speaker data described earlier, and the four speaker multi-repetition data described in the previous section. We compared the entropy of a model trained for a single speaker in the multi-repetition data set to the entropy of a cluster from the 630-speaker set. Only the 18 uninstructed utterances for each speaker were used from the multi-repetition data, because the 630-speaker data were recorded without instruction. The comparison was made with the 10-cluster version of the 630-speaker data so that each cluster would be based on an adequate amount of data. In order to be able to make a fair comparison, it was necessary to compare the entropy of models trained on the same number of speakers, so we sampled the large clusters from the 630 speaker set by randomly choosing a cluster, and then randomly choosing the appropriate number of speakers from the cluster. This was done 1000 times, and the mean entropy of the 18-member clusters were computed. The mean entropy of the 18-member clusters from the 630-speaker data was 8.39, and for a single speaker from the multi-repetition data was 6.86, approximately 18% lower. This suggests the possibility of significant individual speaker adaptation beyond the choosing of appropriate clusters.

DISCUSSION

The studies described in the previous four sections have demonstrated some types of structure in the phonological variation observed in a data set consisting of two sentences (21 words) read by many speakers. In addition, we have shown some types of lexical representations that might be used to capture this structure. Representations were compared by measuring their entropy, or predictive ability. It is assumed that lower entropy can lead to improved recognition performance. In the near future, we intend to test this assumption in a series of recognition performance studies.

The results described above have a number of implications for system design. The first study suggested that a significant advantage in recognition accuracy can be gained by incorporating pronunciation probabilities in a lexical model. The major problem in incorporating such knowledge into large vocabulary systems is finding sufficient amounts of training data to adequately estimate allophone probabilities for the segments of each word in the vocabulary. A possible solution to this problem is to use knowledge of phonological rules, rule groups, and the co-occurrence of allophonic forms to reduce the number of independent probabilities being estimated.

The second study showed co-occurrence relationships between allophonic forms. In addition, an automatic clustering technique was demonstrated that could be used to model this co-occurrence for a data set without explicit knowledge of what these co-occurrences are. This result suggests that lexical representations can be improved by including a small number of sets of word models, each trained on an appropriate cluster of a large data set. When scoring sequences of word pronunciation hypotheses for an utterance, each sequence would only include one set of word model probabilities.

The last two studies suggest methods of adaptation to a new speaker, as well as ongoing adaptation within a session with a single speaker. In figure 7, $H(S|c)$ is shown to be considerably lower than $H(S)$. This suggests that predicting the appropriate cluster for an utterance can reduce entropy considerably by allowing the search to be confined to the model of a single cluster.

The third study, which explored the consistency with which a speaker remains in a cluster, suggests that predicting the cluster for an utterance cannot be achieved solely by speaker adaptation, since a speaker will not stay in a single cluster consistently. However, the third study does suggest that $H(S|c)$ can be approached by choosing an appropriate weighting function over all the clusters, given some experience with a speaker. Furthermore, these results suggest that knowledge of speech rate can be used to improve prediction of the appropriate cluster for an utterance. Ongoing adaptation might be achieved by periodically recomputing the weighting function. We have not explored the question of when, or how often, should this weighting function be recomputed.

The results of the fourth study, comparing intra-speaker to intra-cluster entropy, show greater consistency within a single speaker than within the clusters found in the previous studies. This suggests that speaker adaptation can be improved beyond the choice of clusters by further refinement of model parameters, based on extended experience with a speaker. The major problem with individual speaker adaptation is that model parameters have to be estimated from a small amount of data for the speaker. The advantage of adaptation by cluster choice is that the cluster could be well trained on large amounts of data. The problem of insufficient data for individual speaker adaptation can possibly be handled by exploiting knowledge about phonological rules, rule groups, the co-occurrence of allophonic forms, and implicational rule hierarchies, in order to decrease the number of parameters being estimated, as well as increase the number of samples for each parameter. We intend to explore methods for doing this in future work.

CONCLUSIONS

We have performed a series of four studies, with the following results:

1) Incorporating empirically determined probabilities of allophonic forms into a phonological model can significantly reduce model entropy, and possibly improve recognition accuracy.

2) There is significant co-occurrence of allophonic forms within an utterance, and automatic clustering procedures can be used to compile knowledge of these co-occurrences into a phonological model, without need to explicitly determine what the co-occurrences are. Incorporating these co-occurrences into the phonological model can significantly lower entropy and allow a form of within-utterance adaptation, possibly improving recognition accuracy.

3) Speakers tend to fall into phonological groups. Rapid adaptation techniques might work by choosing either a set of clusters or weighting function over all clusters for a speaker given a small amount of experience with that speaker. Ongoing adaptation may possibly be achieved by periodically rechoosing a cluster set or recomputing the weighting function.

4) Individual speakers vary less than speaker clusters, and therefore, further adaptation to an individual speaker could be useful. This may require the exploitation of knowledge about phonological rules, rule groups, implicational rule hierarchies, and the co-occurrence of allophonic forms.

REFERENCES

[1] Duda, R. and Hart, P., "Pattern Classification and Scene Analysis", John Wiley & Sons, 1973, pp. 225-237

[2] McEliece, R., "The Theory of Information and Coding: A Mathematical Framework for Communication" in "Encyclopedia of Mathematics and its Applications, Volume 3", Rota, G., ed, Addison-Wesley, 1977, pp. 15-34.

LEXICAL ACCESS WITH LATTICE INPUT[1]

Hy Murveit, Mitchel Weintraub, Michael Cohen, Jared Bernstein
Speech Research Program
SRI International
Menlo Park, CA 94025

ABSTRACT

This paper describes alternative approaches to lexical access in the CMU ANGEL speech recognition system. One approach explores scoring alternatives within the framework of the CMU module. In another approach, the asynchronous phonetic hypotheses generated by the acoustic-phonetics module are converted to a directed graph. This graph is compared to a pronunciation dictionary. Performance results for the approaches and the original CMU approach were similar. An error analysis indicates promising directions for further work.

OVERVIEW

A lexical access subsystem can be divided into two major components. One component is a *lexicon*; a data structure that contains a list of words and a representation of the allowable pronunciations of those words. Those pronunciations may have associated probabilities and the probabilities may be dynamic in nature. That is, they may change due to new estimations of speaker-type, speech style, and so on.

The other component is the *search and scoring mechanism*. This compares the output of an acoustic-phonetics module, with the lexicon and determines the word sequence that with highest probability corresponds to those outputs. In doing so, the search and scoring module must take into account the characteristics of the AP output, such as insertion, deletion, and substitution probabilities.

This paper evaluates alternative search and scoring mechanisms in a lexical access module.

GOALS

This is a progress report on work at SRI International in cooperation with Carnegie-Mellon University (CMU) and sponsored by DARPA. SRI is exploring alternative approaches to lexical-access in the framework of a speech recognition system (ANGEL) being developed at CMU [1]. The ANGEL system is designed to recognize a large vocabulary from American English continuous speech. Our goal is to devise an approach to lexical access that both takes advantage of all information available from other knowledge sources in the speech recognition system (particularly the acoustic-phonetic knowledge source), and also is resilient in the face of errors made by those other knowledge sources.

[1]This research was sponsored by Defense Advanced Research Projects Agency Contract N00039-85-C-0302. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

OVERVIEW OF THE CMU LEXICAL-ACCESS MODULE
(circa summer 1986)

A lexico-centric block diagram of the ANGEL speech recognition system is shown in Figure 1. The acoustic-phonetic module [2] outputs a set of phonetic hypotheses (see Figure 2) to the lexical-access module. These lattices contain "firings" that give the estimated probabilities of segments occurring in particular time intervals. However, the relative probability of one firing versus another is not estimated, even if the two firings overlap it in time. This is because the acoustic-phonetic module is made up of a set of independent segment locators and classifiers.

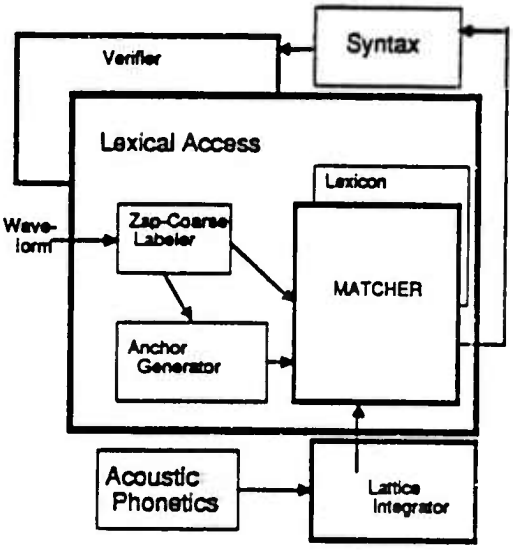


Figure 1.
A lexico-centric block diagram of the ANGEL speech recognition system

The 1986 version of CMU's lexical-access module converted this lattice structure into an *integrated lattice*. The lattice integrator created boundaries where acoustic-phonetic segments began and ended and, in particular, created new boundaries where segments overlapped. It collapsed information from the acoustic-phonetic lattices by combining the probability estimates from overlapping acoustic-phonetic segments to derive likelihoods of the newly created segments. An example of this integrated-lattice data is shown in Figure 2.

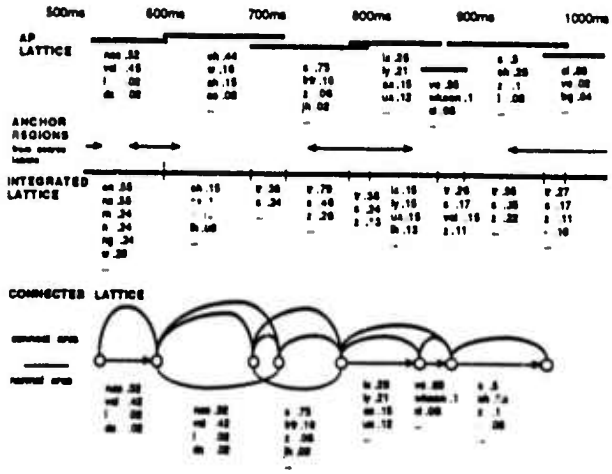


Figure 2.
Acoustic phonetic lexical-access interface data structures

The integrated lattice was matched to a dictionary network that represented all allowed pronunciations of words. The match algorithm allowed one or more integrated-lattice segments to match a particular dictionary segment. A scoring algorithm computed the costs of matching paths in the dictionary and the integrated lattice. The scoring was based on the integrated-lattice likelihoods, comparisons of expected durations of dictionary segments with durations of corresponding concatenated integrated-lattice segments, and comparisons of the integrated-lattice segments with an independent coarse labeling of the input speech (based on the ZAPDASH coarse labeler [3]).

The lexical-access module also computed anchor regions (see Figure 2) from the waveform. Anchor regions define possible syllable and word boundaries as the consonantal regions between vocalic regions specified by the ZAPDASH coarse labeling routine.

The match routine searched for words between all (reasonable) pairs of anchor regions. The match routine did so by looking for patterns in the integrated lattice that matched patterns in its lexicon. The match routine passed matches and their likelihoods to the syntactic module. The syntactic module hypothesized sentences, verifying the word junctures with the lexical-access module.

MODIFICATIONS TO THE CMU SYSTEM

SRI decided to explore a number of alternative scoring and search algorithms to determine how to make the best use of the information contained in the integrated lattice. We limited our search of alternative lexical scoring/search algorithms to variations of the following CMU-lexical-access characteristics.

1. Lexical phonemes were required to begin and end at the boundaries of the integrated lattice.

2. Phoneme scores were weighted by the duration of the phoneme

3. Phoneme scores included penalties if the duration was below or above a preset minimum/maximum duration.

4. Phoneme scores included penalties when their phoneme type did not agree with the coarse-labeling information.

Our goal was to see how well a lexical scoring algorithm could hypothesize words if you knew where the words begin and end. The search of alternative lexical scoring algorithms tried to maximize the rank of the correct word given the known begin and end times of that word. Our best algorithm of this type is described below:

1. Phoneme scores were not weighted by their duration. Therefore, the score of a word candidate was the average of the individual phoneme scores. (In the CMU algorithm, the word score is the duration weighted average of the individual phoneme scores.)

2. Lexical phonemes were allowed to begin or end every 10 msec. A minor degradation in performance was observed when phonemes were required to begin and end at the boundaries of the integrated lattice.

3. The preset minimum/maximum duration constraints were used as hard limits on the allowable duration of a phoneme.

4. The coarse labeling information was not used to modify the phoneme scores.

Evaluation and Testing Data

The matcher and lattice integrator portions of the CMU lexical-access module were compared with the alternative routine described above. These modules were tested on 100 "Electronic Mail" sentences, part of a larger database collected at CMU. In this data, each of 10 speakers said ten sentences composed from a 339 word vocabulary. An example sentence is *"Send a message to Smith at CMU."* The data was hand-labeled at CMU and the outputs of the CMU acoustic-phonetic module and CMU's anchor generator module were sent to SRI. All CMU modules for this study are circa mid 1986. All testing was done on this continuous-speech database in a speaker-independent manner with no grammatical constraints.

The results shown in the tables below are the percent correct words in the text set for the lexical access algorithms given correct anchor regions and hand-set endpoints.

Lexical Access Performance with Hand-Set Endpoints

Rank	*Alternate*	*CMU System Simulation*
Correct	.55	.45
Top 3	.79	.71
Top 10	.94	.88

Lexical Access Performance Using Anchor Regions

Rank	*Alternate*	*CMU Reported Results*
Correct	.30	.32
Top 3	.55	.55
Top 10	.81	.76

As our results show, the performance advantages gained on known endpoints did not translate into an advantage when endpoint intervals were used. However, we believe that this scoring algorithm might result in an improved word accuracy if sentence hypotheses are constructed from the individual word hypotheses. Our current research has therefore been aimed at generating sentence hypotheses.

CONNECTED LATTICES

SRI also explored a modification to the mid-1986 CMU system which eliminated the integrated-lattice component and hypothesized words directly from a structure more similar to the acoustic-phonetic output.

We felt that a significant amount of information was being lost by the integrated-lattice component. For instance, segmentation decisions made by the acoustic-phonetic module were in effect over-ruled by the matcher using the integrated lattice. Similarly, decisions about relative likelihoods of overlapping acoustic-phonetic firings were made by the integrated lattice, when such decisions should have been made by the acoustic-phonetics module.

However, the acoustic-phonetic output without the lattice integrator was not amenable to direct lexical access. Often the string of correct phonetic hypotheses were in the lattice, but either one correct phonetic segment overlapped with the next correct one or was separated from it by a small interval of time. This problem was solved when using the lattice integrator by splitting segments at all overlap points and by allowing single dictionary segments to match a series of integrated lattice segments.

A new representation of the acoustic-phonetic data was chosen (the *connected lattice*) that retains the information in the acoustic-phonetic lattice such as duration of segments and reduces the possible search space by not introducing additional boundaries at segment overlaps.

Conversion to the Connected Lattice

An algorithm that transforms the acoustic-phonetic lattice to the connected lattice works as follows:

The acoustic-phonetic lattices are converted to a simple directed graph, the connected lattice. There are two types of arcs in this graph. AP ARCS are created by replacing each acoustic-phonetic firing with an arc going from a node representing the start time of the firing, to a node representing the end time of the firing. CONNECT ARCS are created between all nodes that have incoming AP arcs and all nodes within 100ms of these nodes that have outgoing AP arcs. Connect arcs are necessary because without them there typically would not be a connected path between the start and end of a sentence.

Output probabilities are assigned to the AP arcs. These probabilities are the product of a vector and a matrix. The vector consists of the probabilities of phonetic segments as assigned by the acoustic-phonetic module in the time interval corresponding to the AP arc (or acoustic-phonetic firing). The matrix is a segment-confusion matrix corresponding to the observed performance of the acoustic-phonetic module. Probabilities are also assigned to the connect arcs in a context dependent manner that makes more reasonable connects more likely. This is described below.

Search of the Connected Lattice

A search is performed to compare the system's lexicon (stored in a pronunciation graph) with the connected lattice. Tuples consisting of the initial lexicon node and all initial nodes in the connected lattice (corresponding to all permissible word starting points given the anchor regions) are placed on a list of active paths. The items in the list are called partial paths. The search algorithm proceeds by taking a partial path off of the list, extending the path in all possible ways[2] (the product of every lexicon arc leaving the lexicon node at the end of the partial path, and every AP or connect arc leaving the connected lattice node at the end of the partial path). These new paths are placed back in the list. Paths that are complete (that end in the end anchor region) are also placed in a list of complete paths.

Partial paths (sets of associations of dictionary segments and connected lattice arcs) are scored as the sum of the log-probabilities of the components of the path. A component probability for an connected lattice is the probability of the dictionary segment in the set of output probabilities of the associated AP arc. The component probability for connect arcs, which have no associated dictionary segment, is a function of the length of the connect arc relative to the lengths of the AP arcs that surround them. For instance two long AP Arcs connected by a short connect arc would have a much higher probability than two short AP arcs connected by a long connect arc.

It is the function of the connect arcs to permit AP arcs to connect reasonably without affecting the score of the paths, however unreasonable sequences of AP arcs are inhibited by the scoring of the connect arcs. The connect arcs also, in effect, lessen the effect of premature segmentation decisions made by the acoustic-phonetic module.

Evaluation

The above algorithm was evaluated using CMU's 100 electronic mail sentences described above. The results are summarized in the tables below.

Lexical Access Performance Using Anchor Regions
(CMU reported results)

Rank	*CMU (324 words)*	*Connected Lattice (240 words)*
correct	.32	.35
Top 3	.55	.52
Top 10	.76	.74

Lexical Access Performance with Hand-Set Endpoints
(CMU system was simulated at SRI)

Rank	*CMU (240 words)*	*Connected Lattice (240 words)*
correct	.45	.63
Top 3	.71	.79
Top 5	.80	.84
Top 10	.88	.92
Top 20	.90	.93

DISCUSSION

The recognition results above show similar performance for the two modules. A closer examination of the data revealed that the connected lattice module tended to have catastrophic errors. Such errors typically occurred when one of the proper segments was deleted by the acoustic-phonetics modules. This version of the connected lattice search algorithm was not equipped to deal with such problems. For instance, of the 17 words not in the top 20 choices for the connected lattice system with hand-set endpoints, 16 were caused by the AP-module's failure to spot one or more segments in a word. One error was due to a speaker's mispronunciation of that word. Of the words that were in the top 6 through 19, the overwhelming majority had the segments there but with low probabilities.

In order to soften the effect of AP deletions, yet continue to take advantage of the acoustic-phonetic data, new connected lattices being designed should include insertion, deletion, and substitution probabilities (separate from phonological insertion, deletion and substitution) for segments based on a model trained with acoustic-phonetic module output. Probabilities for connect arcs will be estimated from similar data, however, in later systems it is hoped that the acoustic-phonetic module will also provide some information about the reasonableness of inter-segment junctures.

[2]The search algorithm does not allow paths to loop, nor can paths have two consecutive connect arcs or begin with or end with connect arcs.

A proposal that describes a new interface between the acoustic-phonetic module and the lexical-access module is included in the appendix to this paper.

REFERENCES

[1] Adams, D. and R. Bisiani, "The Carnegie Mellon University Distributed Speech Recognition System," Speech Technology, March/April 1986 pp 14-23

[2] Cole, R., M. Phillips, B. Brennan, and B. Chigier, "The CMU Phonetic Classification System," Proc. ICASSP, 1986, pp. 2255-57

[3] Gill, G., H. Goldberg, R. Reddy, and B. Yegnanarayana, "A Recursive Segmentation Procedure for Continuous Speech," CMU Technical Report CS-78-134, Computer Science Dept., Carnegie Mellon Univ., Pitts. PA, May 1978

APPENDIX

Proposed New Interface between Acoustic-Phonetics and Lexical Access
(abbreviated form of a memo circulated Fall 1986)

OVERVIEW

The goal of this proposal is to define an interface between the Acoustic-Phonetics (AP) module and the Lexical-Access (LA) module that will improve overall system performance.

The main thesis of this proposal is that some hard decisions (typically segmentation decisions) that are made in the AP module are better left to the LA or even syntax modules, using probabilities assigned by the AP module. For instance, an acoustic-phonetic network that provides probabilities for different segmentations might be used to avoid many of the problems of deleted or split segments, after dictionary and syntactic constraints are applied.

The second thesis of this proposal is that the performance of the AP module should be evaluated in the context of the LA module and vica versa. This implies that to improve system performance a tight feedback loop should be established for developing the two modules. For example, a new network-based AP module should be frequently presented (perhaps even in half-baked form) to the LA group, who should then evaluate word hypotheses, discover specific areas in the AP data as well as in the LA algorithms that need the most improvements, and feed this back to the AP people for further refinements.

GENERAL DISCUSSION

We have come to the conclusions that performance of the top word hypothesis made by the lexical access module is best without a lattice integrator. However, without such an integrator, AP errors are more serious causing a higher percentage of words not to be included in the top 20 words hypothesized by the LA module. The proposed interface between the two modules is expected to help solve this problem.

The following problems are listed in order of seriousness of error (those causing the most problems to those causing the least) based on an analysis of errors made by our current LA algorithms:

1) *PHONE DELETION ERRORS*

The AP module often combines adjacent phones into a single phone with no competing two-phone hypothesis. This can cause a fatal error for word hypothesis routines that do not account for AP segment deletion, since the lexicon only accounts for phonological deletions and for some generalizations about AP deletion. In fact, phone deletion and insertion are the major causes of words missing from the (non-lattice-integrated) word lattice in our experience.

The lattice integrator deals with phone deletion errors by oversegmenting, that is by creating extra boundaries on all overlaps. By creating extra boundaries, the word network can be traversed in cases where it could not be without a lattice integrator. This allows the correct word to be included in the top 20 words hypothesized more often, but often permits incorrect words to achieve better scores than the correct word.

Ideally, the AP module would hypothesize many segmentations, though some might have low probabilities. It is inevitable, however, that some deletions will occur. Therefore, statistics for estimating the probability of deleted phones are necessary for the LA module. First-order statistics such as "the probability that an /ih/ is deleted anywhere" can be computed on a large data base such as the one that will be used to estimate phone confusion probabilities. Second-order statistics, such as probability of deleting an /ih/ after an /iy/ (as in the word "ceeceeing"), may be more desirable. In this example, although the general probability of deleting a vowel may be low, deleting a vowel in the context of another vowel may be much more likely. Higher-order statistics, such as the probability of deleting segments in particular words, may be even more helpful for certain high frequency words.

2) *PHONE INSERTION ERRORS*

The AP module often splits phones or inserts spurious phones which affects the LA module in ways similar to phone deletion. Error statistics can be computed for phone insertion similar to those for phone deletion. Furthermore, a AP post-processor might examine the AP data for consecutive similar segments and creates an additional AP segment if it decides that there is positive probability that these two segments represent one underlying segment.

3) *ANCHOR ERRORS*

The current lexical access algorithm uses anchor regions based on ZAPDASH analysis to propose regions where words may start. Although the claim has been made that 98% of all words are found by this analysis, the boundaries proposed by ZAPDASH are not necessarily even close to the appropriate phone boundaries produced by the AP module. Clearly, anchor generation must be synchronized with the phone alignment! Our lexical-access analyses for systems without lattice integration do not use anchors, but rather hand-marked times corresponding to AP lattice output that may then be "fuzzified."

4) *ERRORS OF PHONE ALIGNMENT*

The proper phones may be located and classified but they may be erroneously aligned. This is shown in the figure below for the word "TV" ==> t iy v iy

Figure 3.
Alignment problems in the AP module

Adjusting the alignment of segments is important when recognizing from current AP lattices. Better alignment can be accomplished by requiring locators for different segments to use common information (such as a coarse labeling of the input speech) when making alignment decisions.

One current SRI lexical access algorithm uses heuristics to solve this problem. This is not the most desirable solution, but in the absence of acoustic-phonetic information to determine the goodness of alignment choices, it is an expedient choice. Overlaps or gaps were penalized more with an increased ratio of the length of the overlap or gap to the sum of the lengths of the segment connected by the overlaps or gaps.

In contrast, the lattice integrator solved the phone-alignment problem by creating new AP boundaries at overlaps and by ignoring gaps if there were no labeled segments in the gap.

Any alignment scheme should aim towards allowing phones to follow each other reasonably, while disallowing incorrect phones with too much overlap or gap. Since absolute decisions can not be made with 100% certainty, alternative segmentations should be provided. When alternative segmentations are provided, probabilities of these segmentations should be estimated as well. It is more appropriate to have the AP module to apply acoustic-phonetic information to evaluate the junctures between segmental hypotheses, than to have the LA module make these decisions.

5) *ERRORS OF PHONE SUBSTITUTION:*

With phone substitution errors, the correct region for a phone is found, but the correct phone is labeled with little or no probability. This can be overcome by characterizing the phone-substitution probabilities of the AP module over a large data base, and using this "phone confusion matrix" in lexical access. This is currently being done by the AP module. Of course, the estimation of $P(phone_i \mid label_j)$, the confusion matrix, should take into account the a priori probability of $label_j$.

PROPOSAL

One possible interface is outlined below. This interface starts with an AP module similar to the current CMU module, but with better phone alignment (perhaps based on the ZAPDASH segmentation scheme). This system, however, also explicitly rates the probabilities of lattice phones following each other which has the effect is of doing "phone-juncture verification" for all nearby phones. Also, the probabilities for merging similar labels are explicitly computed by the AP module. This system is then statistically characterized in terms of phone substitution probabilities, phone deletion probabilities, and phone merger probabilities. All this information is represented in a directed-graph data structure.

INTERFACE SPECIFICATION

A directed graph data structure is output by the AP module with the following characteristics:

1. Each node in the graph represents a particular point in time. There may be several nodes corresponding to the same point in time. This may happen, for instance, if a particular AP event is dependent on another event. For instance, a vowel may be dependent on a following nasal. Then the hypothesis is only connected to things consistent with its hypothesis.

2. There are arcs leaving each node in the graph. These arcs correspond to one of two possible things.

(a) The firing of a locator for a given type of phone (an AP arc). AP arcs always lead to other nodes. (b) The possibility that a locator did not fire for a given phone (an insertion arc). An insertion arc always leads to the node it came from.

3. There are two probabilities associated with each AP arc (above):

(a) the probability that the locator firing corresponding to the arc was valid, P(AP-arc | node), the transition probability for the AP-arc. (b) the set of output probabilities of phones given the AP arc is valid, P(phone | AP-arc).

4. Similarly, there are two probabilities associated with each insertion arc: the probability that any phone can be inserted at the node, which is P(insertion-arc | node); and the probability of a particular phone being inserted at this point given that there was an insertion, which is P(phone | insertion-arc), the output probabilities of the insertion arcs.

5. For a given node, the sum of the transition probabilities for the arcs leaving that node, in other words all the AP arcs plus the insertion arc, should equal 1.0. Similarly, for a given arc, the sum of its output probabilities should be 1.0.

FINAL CONSIDERATIONS

PROBABILITIES AUTOMATICALLY ESTIMATED

All of these probabilities should be automatically estimated from the outputs of the AP module, so that system changes will not require the tweaking of many parameters.

Further, higher order probabilities are desirable when there is adequate data. Thus, for example, the deletion of phone in context (or in word) for high frequency contexts (or words) would be good information. Output statistics showing probabilities of events and the number of events used to estimate these probabilities would be useful to the LA group. With this information, higher order probabilities can be used when there is enough training data to make them reliable, and lower order probabilities can be used otherwise.

ALGORITHM READJUSTMENT WILL BE NECESSARY

Although this proposed interface should ultimately improve performance, there will be some initial problems with it. These problems should be worked out in the context of the lexical-access module. We propose that the AP group initially output both the finish product graph and intermediate statistics that lead to that graph. These include the initial locator/classifier decisions, statistics, such as the confusion matrix, that lead to the ultimate graph, and insertion and deletion statistics. Information on the methods used to assign probabilities to new arcs should also be presented to the lexical access group.

The NBS Fast Formant Tracker
A Progress Report

William J. Majurski and James L. Hieronymus

National Bureau of Standards
Gaithersburg, Maryland 20899

Abstract

This paper presents a progress report on the development of a fast formant tracker. Our present effort is focused on extracting formants 1, 2, 3 and 4 and their amplitudes from voiced speech. Our goal is to produce formants which can reliably drive the recognition of vowels and semivowels in the context of speaker independent continuous speech. The algorithm is based on peak picking and a data reduction technique which analyzes peak frequencies and amplitudes as a function of time. A side product of the tracker is segmentation of voiced regions of speech that shows promise for use in segmenting some phonemic classes. Evaluation of the tracker on 350 utterances from the DARPA Acoustic-Phonetic database indicates that 90% of all phonetic segments are tracked correctly. This is based on visual evaluation of the formant tracks overlayed on spectrograms. Phonemically, a large portion of the errors cluster around /r/. We are in the progress of adding a retroflexion detector developed at NBS (Gengel, Majurski and Hieronymus 1987) to aid the tracker in these areas.

Introduction

This paper gives an overview of our current algorithm for tracking formants in continuous speech. We want to use formants to assist in machine recognition of vowels and semivowels. At the outset of this project no sufficiently acurate formant tracker existed for our use. This algorithm produces formant frequencies and amplitudes for the first four formants.

Our goal was a formant tracker which was accurate, fast and structured to extract as much information from continuous speech as possible. The formant tracker employs a peak-picking algorithm followed by a peak-combination algorithm. The peak-picking algorithm parameterizes peaks by both their frequency and amplitude. The peak-combination algorithm develops initial tracks, called ridges, which are found by combining peaks which are similar in frequency and minimally seperated in time.

Front End Signal Processing

The formant tracker uses pitch synchronous dfts as its signal processing front end. We are currently using a pitch tracker and synchronous dft routines developed at CMU. During voiced speech (areas where pitch tracker fires) a variable width hamming window is centered on the pitch period. The window size is taylored to the pitch period length. This tends to produce spectra in which the formants are represented clearly. For peak picking this appears to be an optimal windowing of the waveform. Including a second pitch period or overlapping two pitch periods in a single analysis window would add components that are out of phase thus diminishing the clarity of the formant structure. Preemphasis of 6 db per octave is used.

Peak Picking

Spectral peaks are selected by locating the negative-going zero-crossings of the first difference of the pitch synchronous spectra. Zero-crossings which occur during negative excursions of the second difference mark the locations of spectral peaks. The pitch synchronous spectra are used unsmoothed.

Each peak is parameterized by its quantized height, frame, and bin. Within a sonorant region the range of peak amplitudes in db is quantized into ten levels by a simple linear function. Frame refers to the pitch period the peak was extracted from. Bin is a frequency measure corresponding the dft bin number (0-127) of the center of the peak. Only peaks in the frequency range 0 - 4000 Hz. are used in the tracker.

Peak Combination - Simple Ridge Construction

The algorithm groups together peaks that will eventually belong to the same formant. In this first pass, the combination-algorithm is very conservative, grouping peaks that are very similar in frequency and very close in time. For each peak in a region of voiced speech, left and right (earlier in time and later in time) neighbors are sought. Only peaks within 2 time frames and 2 frequency bins are considered as neighbors. A simple distance measure is used when multiple choices are available.

The matcher produces groups of linked peaks called ridges. In the simplest case, a steady state vowel, simple ridges represent the formants. This leaves undone only the assignment of ridges to formant slots. More complicated speech patterns require more sophisticated algorithms utilizing more context. A segmentation-algorithm assists here by splitting up speech in places where clear breaks in ridge patterns occur.

Onee a ridge is found most decisions concerning it, use a small number of parameters. Its frequency is parameterized as minimum, maximum and average frequency. Its amplitude relative to other ridges is parameterized as *strength* and *density*. *Strength* is calculated by summing the quantized height of each peak in the ridge. So it is a function of both the relative amplitude of the ridge and the length of the ridge. *Density* is calculated as the average quantized peak height in the ridge. So it is a function of only the relative amplitude of the ridge. In the remainder of this paper, the terms *strength* and *density* refer to these definitons.

Choosing Formants From Ridges

Ridges are assigned to formant slots based on both frequency and strength. Frequency ranges for each of the formants are determined by speaker pitch. In voiced regions the average pitch (1/average pitch period length) is used to declare the region as having low (below 150 Hz.), high (above 170 Hz.) or medium pitch. Formant ranges are then taken from one of 3 tables. For each formant the table contains a range of frequencies that are unique to that formant. It also contains ranges of frequencies for which the formant choice is not clear. A formant with frequencies in one of these ranges is considered ambiguous and its assignment must be determined in the context of the other formant candidates. Seven ranges exist, four for the first four formants and three for the ambiguous areas in between. An assignment algorithm is used to select, from the candidate ridges, four first choice formants and four second choice formants. The second choice formants are used in a later pass of the tracker. In the assigner, each ridge is parameterized by its average frequency and its strength. The top candidates (by strength) from each of the seven ranges are collected and formant assignments made - stongest ridge first. Ridges in ambiguous frequency ranges contend for two formant slots.

Formant Frequency Ranges*

	Low Pitch		Medium Pitch		High Pitch	
Range**	From Freq	To Freq	From Freq	To Freq	From Freq	To Freq
f1	100	700	140	825	175	1000
f12	700	1200	825	1250	1000	1300
f2	1200	1500	1250	1630	1300	1775
f23	1500	2500	1630	2850	1775	3000
f3	2500	2800	2850	3200	3000	3500
f31	2800	3200	3200	3600	3500	3800
f4	3200	4000	3600	4200	3800	4400

* Uses average ridge frequency

** f1,f2,f3,f4 refer to frequency ranges unique to that formant.
f12 refers to frequency ranges which could hold f1 or f2.
f23 refers to frequency ranges which could hold f2 or f3
f34 refers to frequency ranges which could hold f3 or f4

Figure 1 - Pitch dependent, formant frequency ranges used in formant slot assignments. Low pitch is below 150 Hz. High pitch is above 170 Hz.

Basic Segmentation

The segmenter subdivides voiced regions in a way that assists in the assignment of ridges to formant slots. For some phoneme groups, this corresponds roughly to a phonemic segmentation. Specifically, it attempts to segment nasals, voiced stops, voiced fricatives, and flaps, from the surrounding voiced areas. We have found these locations to harbor a majority of the discontinuities in formant frequency.

The segmenter uses as input a select group of ridges called *important* ridges. Ridges are selected by there dominance in a region, using several criteria.

Ridges that occupy one of the first three formant slots (first and second choice) are used; although, in most cases the second choice slots are empty. Other ridges may be included based on their density measure or on a measure called *overlap*. The density measure is used to include ridges which have very high amplitudes but do not have sufficient duration to win a formant slot. The *overlap* provision finds ridges which (1) overlap first choice ridges in time, (2) are minimally seperated from the first choice ridge during the overlap and (3) have a higher density during the overlap. This *overlap* provision thus, finds places where the initial peak matcher may have failed to follow the formant correctly.

Segmentation clues are extracted from the *important* ridges. The begining and end frame of each ridge is labeled as a clue. Sudden changes in amplitude are labeled. Sudden changes in F1 frequency are labeled.

Segmentation clues associated with F1 tend to mark obstruent boundries such as nasals, flaps and voiced stops. Since these phonemic events tend to harbor formant discontinuities, a segment boundry is created for this type of clue without further confirmation. Segmentation clues associated with higher formants are less reliable. They involve more variation in frequency and amplitude. They also are more heavily influenced by neighboring frication. Segment boundries are only established where two or more clues are found.

Recursive Segmentation

The segmenter drives the search for formants recursively. Each voiced region is initially labeled as a single segment. Basic ridges and important ridges are recomputed and new segment boundries are searched for. New boundries cause the segmenter to subdivide and operate on each resulting piece. The process ends when no further subdivisions can be found.

Use of Bark Scaled Spectra

An attempt was made to use Bark scaled spectra as input to the formant tracker. (Seneff 1986) We had hoped it would help in tracking diffuse upper formants and upper formants influenced by fricatives. While Bark scaling did help in these two cases it hurt in ways we were less willing to cope with. Upper formants that are close in frequency to start with, such as high front vowels, and retroflexion, tended to be merged. We found merges very difficult to handle. Therefore we no longer use Bark scaling of the spectra for formant tracking.

Performance Analysis

The CMU and NBS formant trackers were run on a subset of 350 sentences of the DARPA Acoustic-Phonetic database. The formants were overlayed on spectrograms and examined. Only the voiced phonemes were considered as valid segments for statistics. An error was counted if any of the three formants were not within the dark bands on the spectrogram or if the wrong formant was assigned. In total there were 6716 segments examined. The results of this analysis are shown in Figure 2.

Formant Tracking Errors				
Formant	Sex	Set	CMU	NBS
All	m	u1	8%	9%
	f	u1	25%	7%
	m	v1	8%	11%
	f	v1	31%	14%
	m	v2	9%	10%
	f	v2	26%	12%
F1	m	u1	1%	0%
	f	u1	5%	0%
	m	v1	1%	1%
	f	v1	4%	2%
	m	v2	1%	1%
	f	v2	4%	2%
F2	m	u1	5%	5%
	f	u1	20%	4%
	m	v1	5%	6%
	f	v1	23%	8%
	m	v2	7%	6%
	f	v2	20%	7%
F3	m	u1	4%	6%
	f	u1	12%	3%
	m	v1	4%	9%
	f	v1	20%	9%
	m	v2	6%	7%
	f	v2	20%	9%

Figure 2 - Percent errors in voiced phonemes. Performance on 350 sentences from the DARPA Acoustic-Phonetic Data Base.

Upcoming Changes

Development of the algorithm is not yet complete. Performance in several areas can be improved. We have learned much from our first pass at creating a segmenter based on formant data. We will soon start a complete rewrite of the segmenter which will use many new rules and clues. An evaluation and more complete description will be published later. An energy based retroflex detector(Gengel, Majurski and Hieronymus), already running in our lab will be used to help make decisions during /r/, where F2 - F3 merges are common, during retroflexed vowels. Since this algorithm is based on formant continuity, tracking the high frequency formants during and near fricatives is a special problem also requiring solution.

Implementation

The peak picker and formant tracker are written in C and run on both Unix and the Symbolics Lisp Machine using the Zeta-Soft C cross compiler. All development work on the tracker was done on the Lisp Machine using Spire. The Unix version has been delivered to CMU to be evaluated for use in their system.

Execution speed on our Vax/750 averages 50 seconds per utterance. On CMU's Vax/780 it averages 20 seconds per utterance. This implementation uses fixed point arithmetic only.

Conclusion

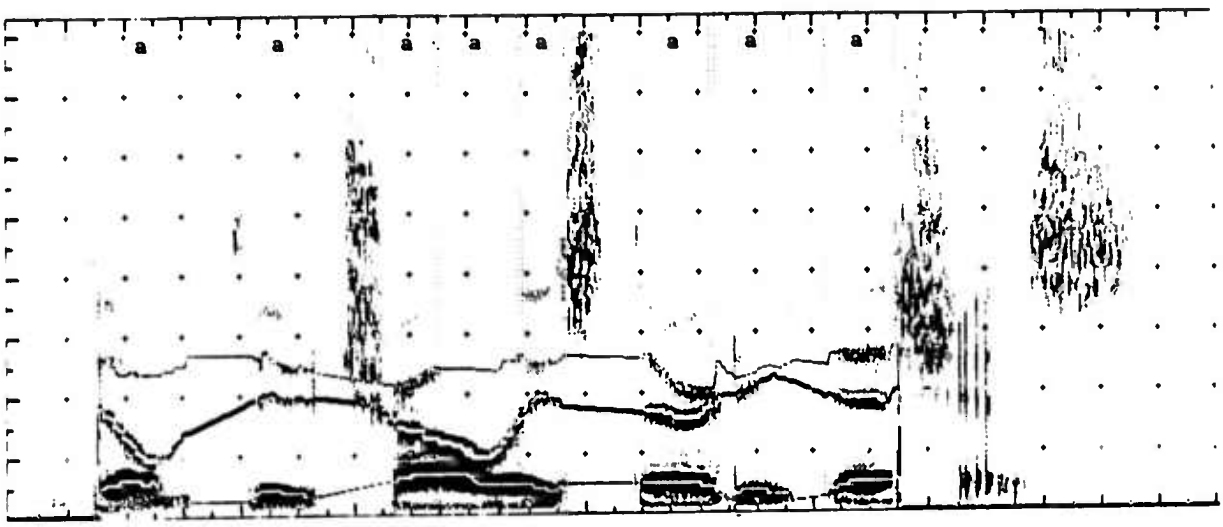
We have presented an overview of our current algorithm for tracking formants in continuous speech. We have described the general algorithm for tracking and for segmenting. The segmenter is currently in a rudimentry state but we believe it holds much promise for segmenting voiced speech. Since we believe that formant analysis is critical to the paramaterization of continuous speech. We will continue working to improve the performance of this algorithm.

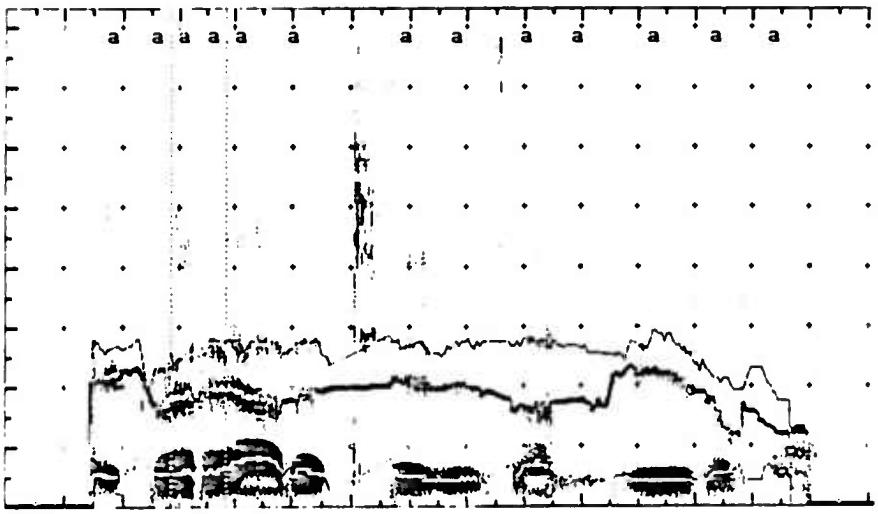
References

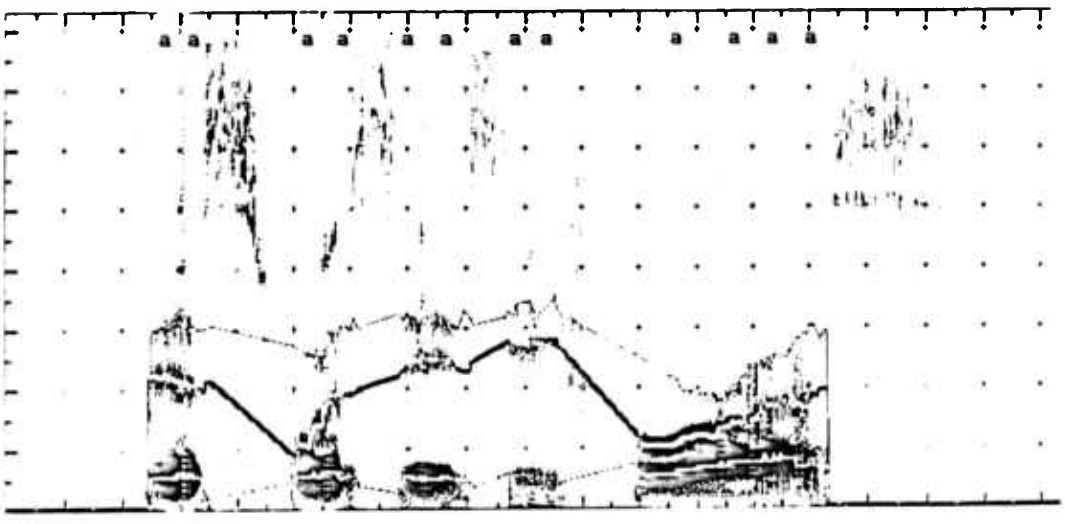
Gengel, R.W., Majurski, W.J., and Hieronymus, J.L., (1987). *An Energy-Ratio Based "Retroflexion" Detector: Current Status and Performance*, these Proceedings.

Seneff, S. (1986), *An Auditory-Based Speech Recognition Strategy: Application to Speaker-Independent Vowel Recognition*, Proceedings - Speech Recognition Workshop, February 1986, Palo Alto, California.

Cyphers, D.S., Kassel, R., Kaufman, D., Leung H., Randolph, M., Seneff, S., Unverferth, J. III, Wilson, T., Zue, V. (1986), *The Development of Speech Research Tools On Mit's Lisp Machine-Based Workstations*, Proceedings - Speech Recognition Workshop, February 1986, Palo Alto, California.







An Energy-Ratio Based "Retroflexion" Detector: Current Status and Performance

Roy W. Gengel, William J. Majurski and James L. Hieronymus

National Bureau of Standards
Gaithersburg, Maryland 20899

Abstract

Results are described for a "retroflexion" detector designed to locate the acoustic manifestations of /r/, /ʃ/, and /ʒ/ in the continuous speech of male speakers. A SEARCH analysis indicates 89 percent correct detections for a false alarm rate of 14 percent. Missed retroflexed tokens fall into three groups: (1) low F4, which is also close to F3; (2) "diminished retroflexion," attributed to a phonotactic rule; (3) F2 and F3 are relatively high and straddle or exceed the specified cutoff for male retroflexion.

Introduction

We are developing in automatic "retroflexion" detector based on energy ratios. It is designed to locate the acoustic manifestations of the phonemes /r/, /ʃ/, and /ʒ/ in continuous speech. The detector will not be described in great detail here. It is based on the idea that a third formant below 2000 Hz for males and 2300 Hz for females is a correlate of retroflexion. A set of energy ratios has been formulated and tested to detect this event. In its current form, it is designed to detect waveform segments that generally meet the following three criteria:

1) relatively high energy between 1000-2000 Hz coupled with relatively low energy between 2000-3000 Hz (Energy Diff 6);

2) relatively high energy between 1400-2000 Hz coupled with relatively low energy between 2000-3000 Hz (Energy Diff 20);

3) relatively high energy between 120-1100 Hz coupled with relatively low energy between 2200-2800 Hz (Energy Sum 4+5).

The current version of the detector, called Energy Sum R, was developed for males voices. A modified version of the detector is also being developed for use with female speakers. Only analyses based on male voices will be reported here.

Figure 1 shows analyses of two sentences from the data base that may be used as "canonical" representations of /r,ʃ,ʒ/, and can be used to illustrate the underlying logic of Energy Sum R. Note that when the criteria are closely met, the peak in the Energy Sum R display is relatively high. This generally occurs (1) during production of /r,ʃ,ʒ/, (2) sometimes, during production of certain other phonemes coarticulated with /r,ʃ,ʒ/, and (3) sometimes, during transitions of other phonemes (see /ɑ/, /ɜ/ and /k/.)

Corpus and Method

The corpus used for analysis consisted of sentences in a subset of the DARPA Acoustic Phonetic Data Base that are spoken by males speakers. There were 126 sentences which contain a total of 4465 tokens. The breakdown of tokens according to number per phoneme is shown in the upper portion of Figure 2.

In order to perform the test we used the SEARCH Program developed at MIT (Randolph, 1986). The phonetic labels used were the labels provided by the MIT Group.

The SEARCH program was set to find all labeled phonemes which contained values of Energy Sum R greater than a specified threshold value. Thus, included in this search are all phonemes having Energy Sum R values above threshold, regardless of how much or how little was the value above threshold, or how much or how little of the segment contained the suprathreshold value.

Results

The results of one SEARCH analysis is shown in the lower portion of Figure 2. Max Energy Sum 30 represents an arbitrary, but reasonable threshold; reasonable in its trade-off between correct detections and false alarms.

Note that the analysis reduces the corpus to be examined further to 20 percent of the original. Contained therein are 89 percent of the target phonemes /r,ʃ,ʒ/. These "retroflexed" phonemes comprise 32 percent of the reduced corpus.

In the remainder of this paper, we describe some characteristics of the /r,ʃ,ʒ/ tokens that were "missed" by the combined detector. In a companion paper, we describe some of the "false alarms" due to coarticulation effects

and other phenomena.

Thirty four "retroflexed" tokens were "missed" by the analysis: 25 /r/s, 4 /ʧ/s, and 5 /ʤ/s. These 34 tokens were further analyzed to determine whether they possessed characteristics different from the 279 tokens that were correctly detected.

We call the "normal" retroflexed token, one in which the energy in F1, F2, and F3 is located below 2000 Hz, while F4 remains above 3000 Hz. (See Figure 1.) In 15 of the 25 "missed" /r/ tokens, F4 dropped below 3000 Hz. This relatively low frequency F4 caused the denominators of the energy ratios used to calculate Energy Sum R, to become large. This, in turn, resulted in low values of ES R that did not exceed the threshold for "retroflexion" detection. Figure 3 shows some dramatic examples of F4 downward movement in parallel with F3. However, the movement is not always in parallel with F3 nor as dramatic. The characteristics of both F3 and F4 of the adjacent phoneme determine, in part, the type of F3 and F4 movement into the retroflexed waveform region. Presumably, these tokens show "retroflexed alveolar articulation," as contrasted to "retroflexed palatal articulation" (Fant, p.28, 1973). The former are also referred to as "r_1=voiced, continuant, apical" (op.cit., p. 63). This phenomenon requires more detailed investigation since it is not restricted merely to the 15 tokens just mentioned. It also occurred in two "missed" /ʧ/ and in three "missed" /ʤ/ tokens, as well as in tokens where Energy Sum R exceeds threshold during a portion of its waveform duration. The latter condition can be seen clearly in the upper portion of Figure 1 where F4 parallels the diminution in the amplitude of Energy Sum R. (We are currently developing an F4 tracking program to more efficiently identify this phenomenon.)

A second phenomenon which is evident in six "missed" tokens we have tentatively labeled as "diminished retroflexion." This occurred in five /r/s, and in one /ʧ/. It is evidently due to a phonotactic rule employed by some persons living along the eastern seaboard that states: When /r/ (or other retroflexed token) is preceded by a vowel, delete the /r/. This rule is also used in areas of Great Britian (Bristow, 1984) and in some variations of Black Dialect. Acoustically, it is manifested by a relatively high F3 (above 2000 Hz), and a relatively low F2 (below about 1400 Hz). Perceptually, it does seem to differ from the preceding vowel, at least to the untrained ear. We recommend it be given a different token-label, since it is not a retroflexed phoneme.

Finally, a third phenomenon accounts for the remaining "missed" tokens. In these instances, (five /r/s, two /ʒ/s, and one /ʤ/) both F2 and F3 are relatively high: i.e., they straddle, or are above, the 2000 Hz criterion for a male retroflexed token. These tokens are in the frequency regions that we presume are more usual for female voices (or higher pitched voices generally). Nevertheless, they currently are classified as "true misses." However, we are relatively confident they will be detected when the modified version of Energy Sum R, designed specifically to analyze higher pitched voices, is fully developed.

In conclusion, the current retroflexion detector seems to offer great promise. When the "diminished" retroflexion tokens are deleted from the target corpus, the correct detection rate is about 91 percent. Assuming the successful incorporation of an F4 tracking program into the present analysis scheme, the "low F4 misses" should also be detected. Then the hit rate will reach about 97 percent. Finally, the modified Energy Sum R (F) detector might detect the remaining three percent of "missed" retroflex tokens, thereby bringing overall performance to near 100 percent correct detection. This would be its performance capability tempered by a current estimated false alarm rate of about 14 percent (572 incorrect detections/4158 nonretroflex tokens).

Acknowledgement

This work was supported, in part, by DARPA Contract N0003984PD41304.

REFERENCES

Fant, G., (1973). "Descriptive analysis of the acoustic aspects of speech," in: Speech Sounds and Features, MIT Press,Cambridge.

Randolph, M. A., (1986). Described in: "The development of speech research tools on MIT's Lisp Machine-based workstations," by: Cyphers, et. al., Proceedings, Speech Recognition Workshop, DARPA, Palo Alto, CA.

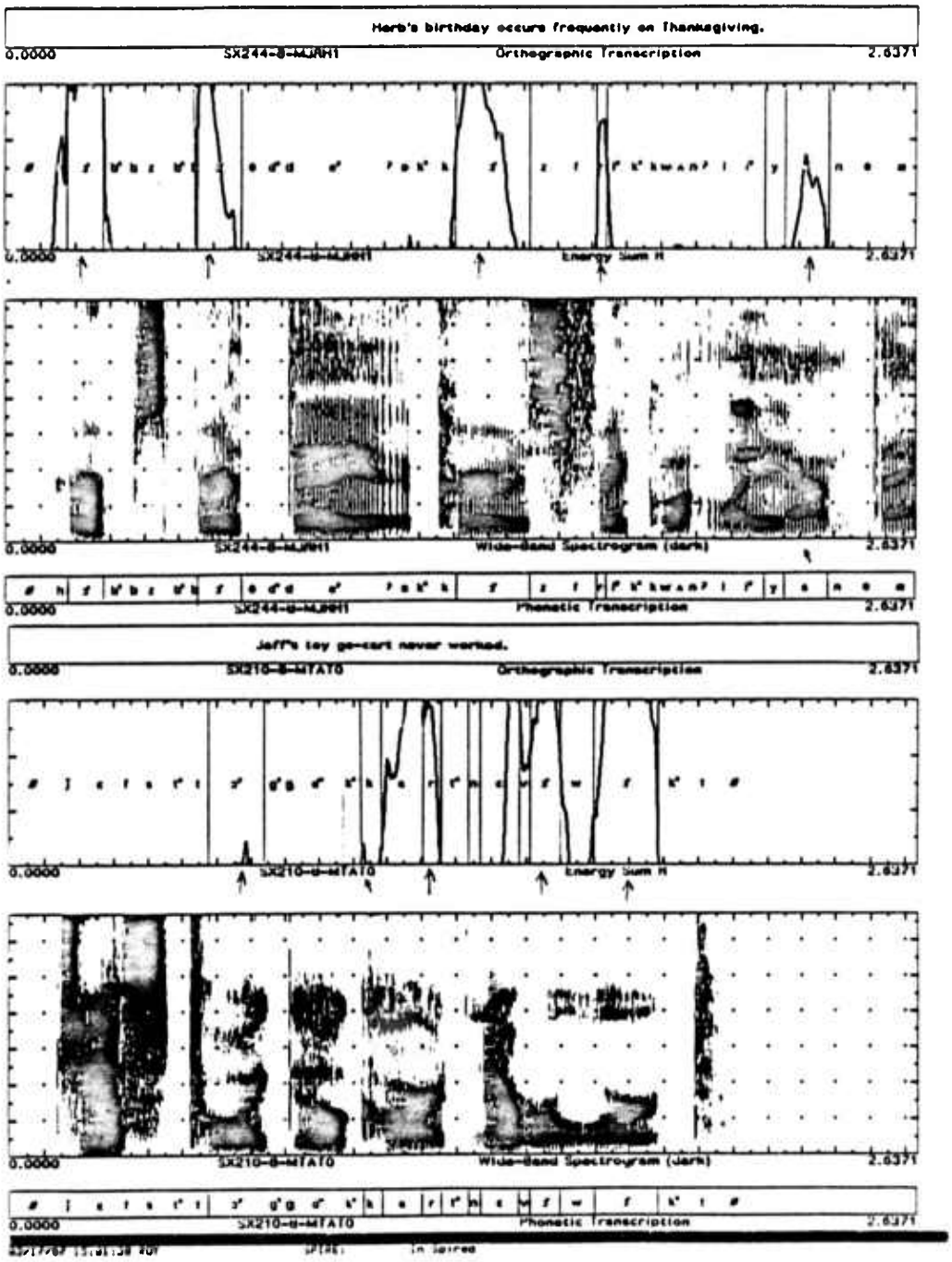


Figure 1. Two examples of the output of Energy Sum R indicating retroflexion of /r/ allophones and coarticulated neighboring phonemes.

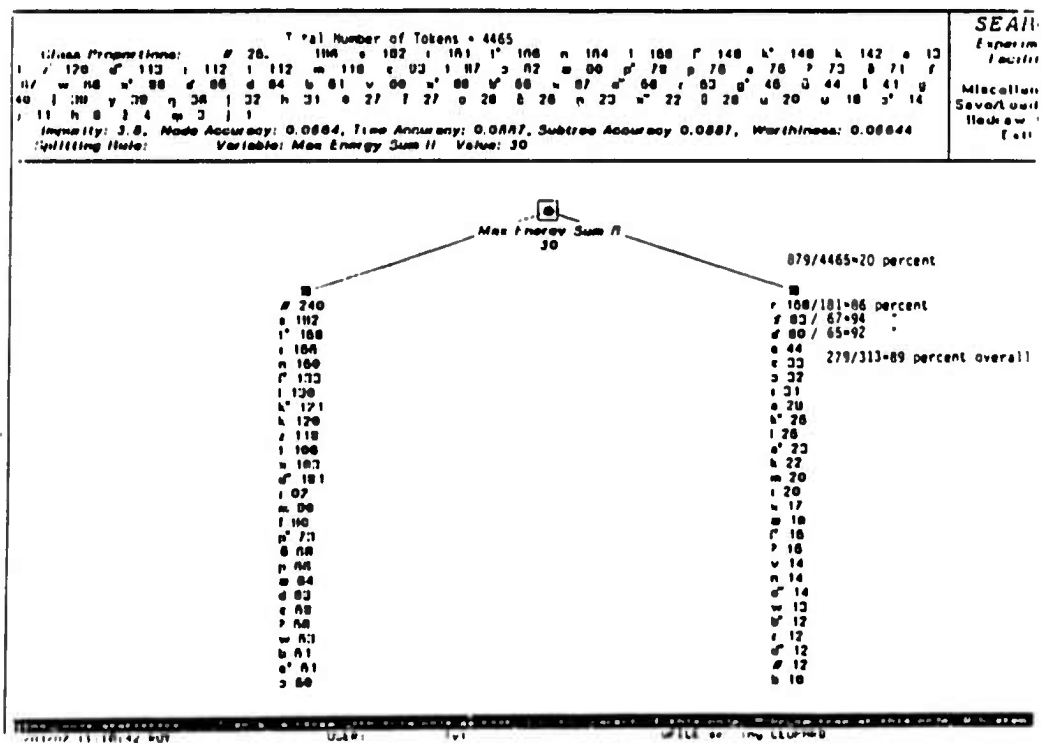


Figure 2. Results of a SEARCH analysis based on a threshold value of 30 for Max Energy Sum R. See text for explanation.

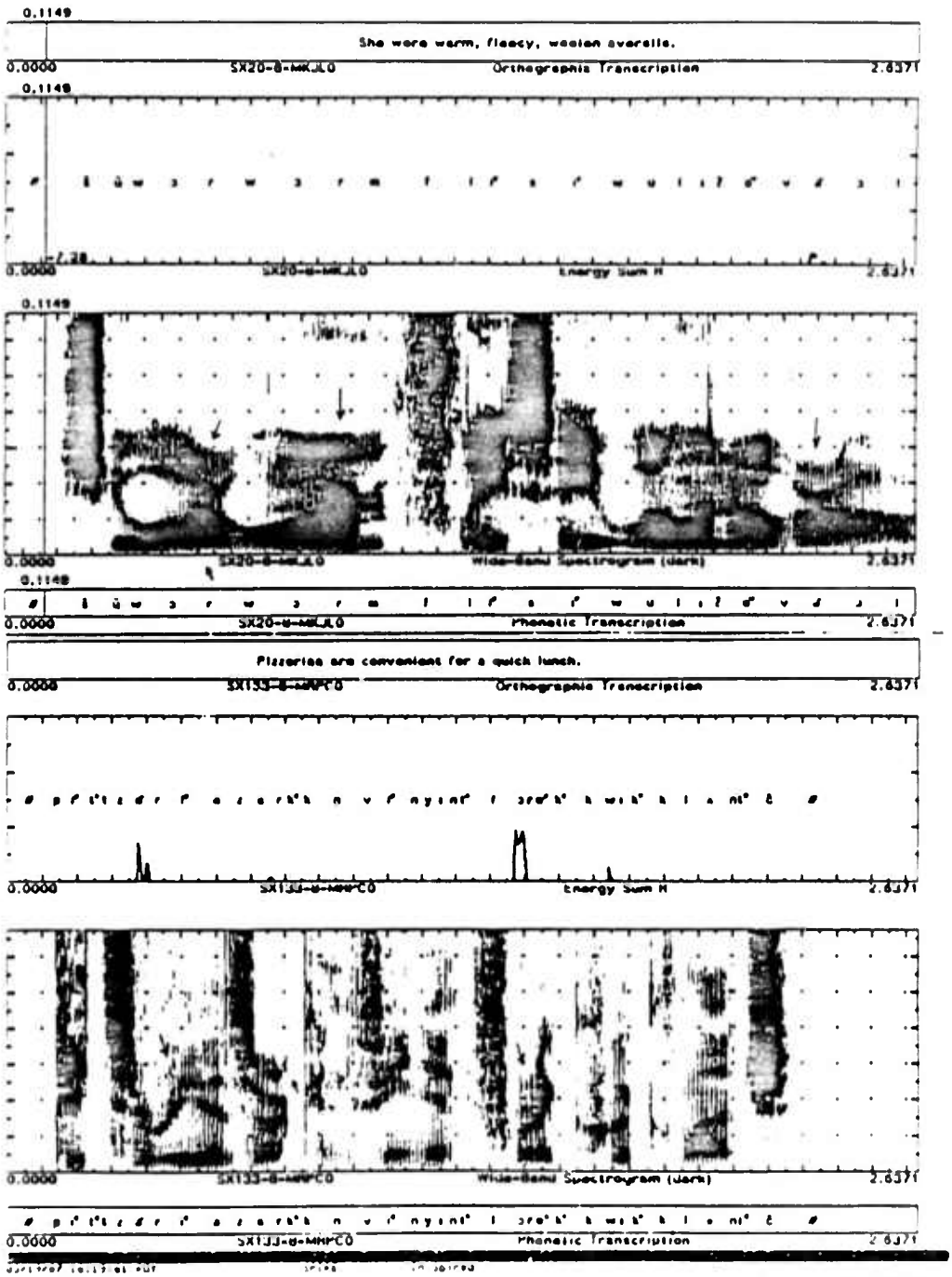


Figure 3. Examples of low F4 during retroflexion. See text for explanation.

An Analysis of Some Coarticulatory Effects of /r/ on Preceding Vowels: Initial Findings

Roy W. Gengel, William J. Majurski and James L. Hieronymus

National Bureau of Standards
Gaithersburg, Maryland 20899

Abstract

SEARCH and a retroflexion detector identified vowel-tokens that (1) were classified as retroflexed, and (2) shared a boundary with /r/. The vowels /ɑ/, /ɛ/, /ɔ/, and /ɪ/, among others, show strong coarticulatory effects. Data supporting this conclusion are presented.

Introduction

A pilot study of coarticulation effects in vowels due to post-vocalic /r/, was made for a subset of the DARPA Acoustic Phonetic Data Base. Using the SEARCH Program, the technique was to find where the retroflexion detector fired inside vowel- tokens that shared a boundary with /r/. We have found the vowels /ɑ/, /ɛ/, /ɔ/, and /ɪ/ are among the labeled nonretroflexed tokens that are strongly retroflexed. (See Figure 2, Gengel, Majurski and Hieronymus, these Proceedings.) We conclude that strong retroflexion is indeed present in these tokens; i.e., F3 is below 2000 Hz, for various portions of their total duration. Histograms of coarticulation extent are presented below.

Method

The corpus of sentences used in this analysis was the same sentences described in Gengel, Majurski and Hieronymus (1987, these Proceedings). However, in order to increase sample size, additional sentences from the DARPA Acoustic Phonetic Data Base were also included.

Figure 1 shows the layout used for analysis. The displays are an Original Waveform Window, a Wide Band Spectrogram, a Wideband Spectral Slice, an LPC Spectral Slice, and a Phonetic Transcription Window, all of which are part of MIT-Spire (Cyphers, 1985); a Vax Psdft Spectral Slice, a display of the CMU-Darpa-System- output; and F2 and F3 formant tracks, a part of the NBS system developed by Majurski and Hieronymus (1987, these Proceedings). The goal is to determine the time in the vowel preceeding /r/ where F3 drops to a value of 2000 Hz or less. As the figure shows, there is often good agreement among the various indicators, as to the frequency of F3. (When there is not, the value determined by the Wideband Spectral Slice Window is used.) When the 2000 Hz F3 pitch period has been located, the cursor in the Original Waveform Window is automatically aligned in the same time frame in the Phonetic Transcription Window. We then measure to determine whether the cursor is in the first, second, third or fourth quartile of the vowel (or whether it is actually in the token even preceeding the vowel; or alternatively, whether it is within the /r/-token boundary itself). Thus, for example, in the top panel of Figure 1, F3 is below 2000 Hz in the /w/ that preceeds the /ɛ/ that preceeds the /r/; i.e., a relatively long coarticulation effect. And in the lower panel, F3 drops below 2000 Hz in the second quartile of the vowel preceeding /r/: i.e., a relatively shorter coarticulation effect.

Results

A summary of the retroflexion analysis is shown in Figure 2. Note that the duration of the vowel preceding /r/ has been divided into quartiles. As the quartile value increases from one to four, the longer is F3 below 2000 Hz prior to the phonemic (perceptual) onset of /r/; and thus, the longer is the duration of the coarticulatory manifestation of retroflexion. (Recall Figure 1.)

Note that most of the sampled vowels show the coarticulation effect: 94 % for /ɑr/, 94% for /ɛr/, 83% for /ɔr/, and 88% for /ɪr/. The duration of the coarticulation effect, for all four vowels varies from relatively short (first quartile) to relatively long. For the /ɑr/ coarticulated tokens, 61 percent of the /ɑ/ durations are retroflexed for over half of their total durations: for /ɛr/, similarly, 41 percent of the /ɛ/ durations are retroflexed for over half their total durations.

From the amount of data analyzed so far it is difficult to reliably fit a gaussian to the histogram data. So the reliable means and variances of the coarticulation durations will be determined in subsequent work.

Based on these initial findings, we conclude that many of the retroflexion "errors" signaled by the detector are not errors but rather reflect the effect of coarticulation. For example, further analysis of the 44 /ɑ/ "errors" (Figure 2, op. cit.), indicate that 25 /ɑ/s preceded /r/, 6 followed /r/, and 13 were not articulated in an /r/ environment. The 25 pre-/r/s and the 6 post-/r/s all showed coarticulation effects. The post-/r/ effects were small, never beyond the first quartile. The 13 "true error" detections have not yet been analyzed fully. However, three are

associated with /ɑ k$^{\alpha}$/ coarticulation, the "velar pinch" described by Zue, (1986), wherein F2 and F3 in /ɑ/, and other, front vowels, "pinch" together at the /k$^{\alpha}$/ boundary.

These strong coarticulations occur across word boundaries as well as within words. Therefore, it is important that the effect of /r/ on nearby vowels be taken into account in the DARPA Speech Recognition Systems.

Acknowledgement

This work was supported, in part, by DARPA Contract N0003984PD41304.

References

Gengel, R.W., Majurski, W. J., and Hieronymus, J.L., (1987). "An energy-ratio based "retroflexion" detector: Current status and performance," these Proceedings.

Majurski, W. J., and Hieronymus, J.L., (1987). "The NBS fast formant tracker: A progress report," these Proceedings.

Zue, V. (1986). "Speech spectrogram reading: An acoustic study of English words and sentences," Workshop Notes, MIT, Cambridge.

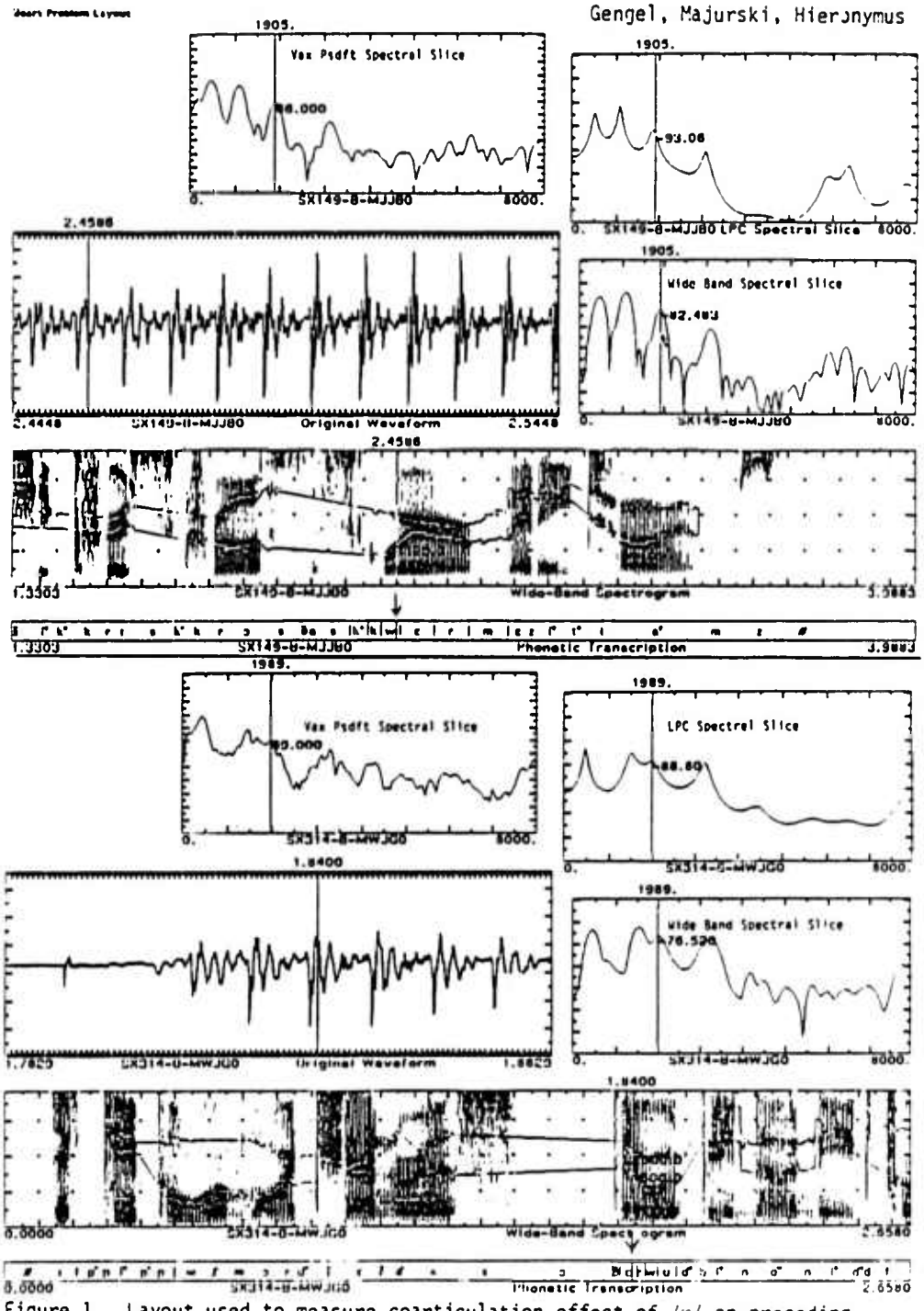


Figure 1. Layout used to measure coarticulation effect of /r/ on preceding vowel. See text for explanation.

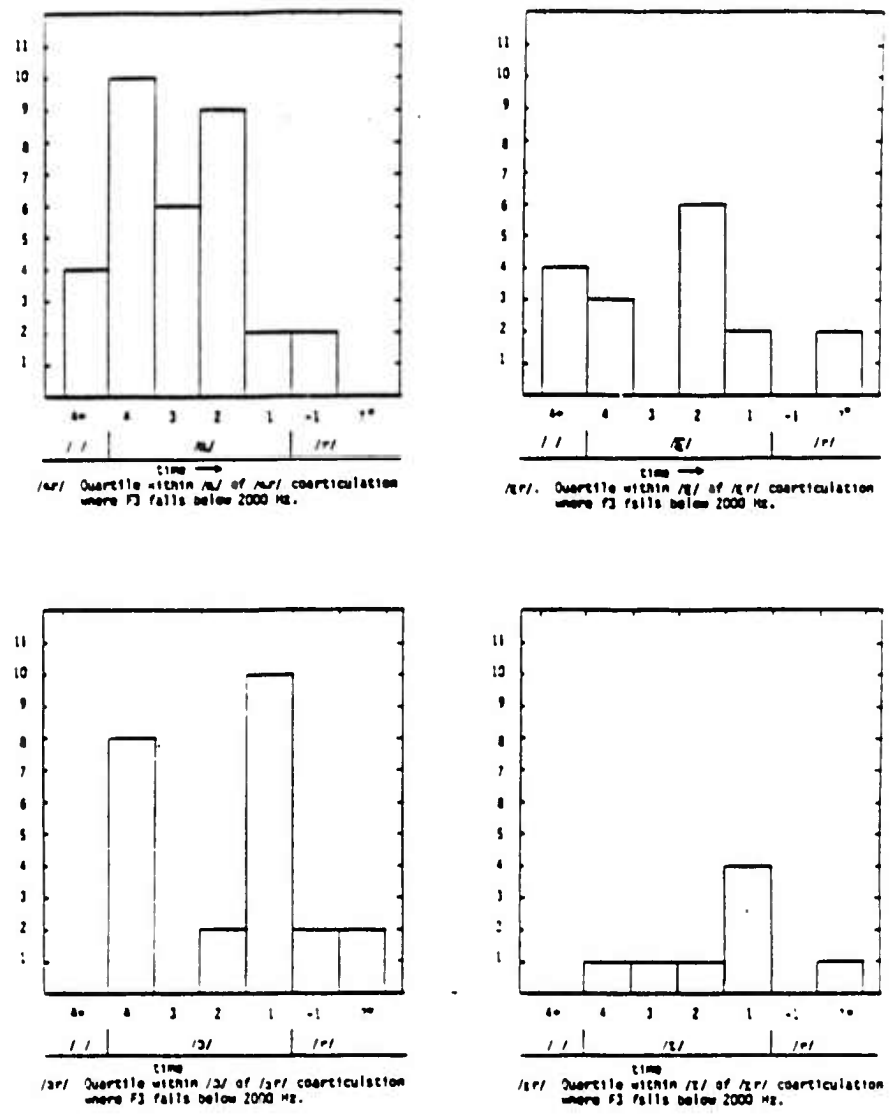


Figure 2. Some coarticulatory effects of /r/ on a preceding vowel. Shown is the duration-quartile within a vowel preceding /r/ where F3 drops below 2000 Hz.

* Indicates that F3 did not drop below 2000 Hz in either /r/ o the preceding vowel.

TEST PROCEDURES

FOR THE

MARCH 1987 DARPA BENCHMARK TESTS

David S. Pallett

Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, MD 20899

ABSTRACT

This paper describes test procedures that were to be used in conducting benchmark performance tests prior to the March 1987 DARPA Meeting. These tests were to be conducted using selected speech database material and input from "live talkers", as described in a companion paper.

INTRODUCTION

At the Fall 1986 DARPA Speech Recognition Meeting, plans were discussed for implementing benchmark tests using the Task Domain Speech Database. There was additional discussion of the desirability of developing and implementing "live tests" using speech material provided by speakers at the contractors' facilities, emulating in some sense the process of inputting speech material during a demonstration of real-time performance. Following the Fall Meeting, the Task Domain Speech Database was recorded at TI and significant portions of it were made available for system development and training purposes through NBS to both CMU and BBN. Another portion was selected for use in implementing these benchmark tests [1], and this test material was distributed to CMU and BBN during the last week of February, 1987. This paper outlines test procedures to be used to implement these tests prior to the March 1987 Meeting.

A number of informal documents have circulated within the DARPA Speech Recognition community that outline proposed test procedures. A Strategic Computing draft document dated Dec. 6, 1985 [2] identified key issues in some detail. Portions of this were heavily annotated and distributed to several sites during June 1986 and were the subject of discussions involving the author and representatives of CMU, BBN, Dragon Systems, MIT and TI during visits during June and early July 1986. These discussions were valuable in developing an outline of benchmark test procedures [3] that was discussed at the Fall 1986 DARPA Meeting, and which was structured after a model for performance assessment tests outlined in an earlier NBS publication [4]. Thus the present proposed test procedure represents the most recent and specifically focussed in a series of documents outlining test procedures for the DARPA Speech Recognition Program.

EXPERIMENTAL DESIGN

There were to be two distinct types of tests conducted prior to the March 1987 DARPA meeting:

(1) Tests based on use of a subset of the Task Domain (Resource Management) Development Test Set Speech Database. This subset was to include use of 100 sentence utterances in either the Speaker Independent or Speaker Dependent portions of the database. The process of selecting speakers and the specific utterances is described in Reference [1]. In each case, there was considerable freedom to choose system-dependent factors such as the amount of training material for Speaker Dependent technology and the most appropriate grammar. All of the 100 specified test sentences were to be processed and reported on at the meeting. "Spell-mode" material (spelled-out representations of the letter strings for items in the lexicon) was available for use, but it processing this material was not required.

These sentence utterances were to be processed both with and without the use of imposed grammars. In the case of using no grammar, the perplexity is essentially to be nominally 1000. Comparable detailed results are to be reported for both conditions. No other parameters are to be changed for these comparitive tests.

Optionally, the same data may be processed using the "rapid adaptation" sentences for system adaptation. There is to be no use of adaptation during processing of the test material.

(2) Tests based on input provided from "live talkers". The test talkers visited both CMU and BBN prior to the March meeting. Each of the talkers spoke the "rapid adaptation" sentences and read a script containing 30 sentences drawn from the task domain sentence corpus. Data derived from the input from "live talkers" was to be analyzed and reported on at the March meeting.

LIVE TEST PROTOCOL

The microphone was to be the same as that used at TI for the Resource Management database, the Sennheiser HMD 414-6. This is a headset-mounted noise cancelling microphone similar to the Shure SM-10 family of microphones. The headset is a supra-aural headset that allows the subject to be aware of nearby conversation or instructions for prompting. The test environment was to be a conference room or computer lab. There was to be no background speech at the time the test material is provided. Test utterances could be rejected (and the subject asked to repeat the sentence) if in the judgement of the person(s) administering the tests there was some noise artifact (e.g. coughs or paper-shuffling noises) or severe mis-articulation of the test sentence. Evidence of this could be obtained by play-back of the digitized utterance.

For systems that require time to develop speaker-adaptive models, the subjects were to provide the 10 "rapid adaptation" sentences prior to the tests (e.g. the evening prior to the tests).

For one of the speakers, the 30 test sentences were to be read in and processing (automatic recognition) could take place "off-line". For the other two speakers, the test sentences were to be read in, one at a time, waiting for the system to recognize each sentence before proceeding to the next sentence. At the end of 30 minutes, if all 30 sentences had not been read in and recognized, the remaining sentences were to be read in for "off-line" processing. In practice, only three to five sentences were recognized interactively within the 30 minute period, and the remaining sentences were then read in. The elapsed time for each speaker providing the test material in this manner was typically 45 minutes. If requested, each speaker was to read in 10 words randomly chosen from the "spellmode" vocabulary subset.

PROCESSING OF LIVE INPUT

The systems were to process the test material in a manner similar to that used for the Resource Management database test material. Statistics comparable to those for the 100 sentence subsets were to be prepared and reported on at the March meeting.

ADAPTATION

Although the use of the "rapid adaptation" sentences was to be permitted, it appears that the only use made of the rapid adaptation sentences was in adapting the Speaker Dependent system at BBN for the "live test" speakers.

There was to be no use of any of the test material to enroll, adapt or to optimize system performance for the test material through repeated analyses and re-use of the test material. Intended allowable exceptions to this prohibition against re-use of the test material include demonstrating the effects of using different grammars, different strategies for enrollment, different algorithms for auditory modelling, acoustic-phonetic feature extraction, different HMM techniques, system architectures, etc. It is recognized that the breadth of these exceptions in effect limit the future use of this test material, since such extensive use of test material to demonstrate parametric effects constitutes training on test material.

Since a finite set of task domain sentences was developed at BBN, and the entire corpus of task domain sentences was made available to both CMU and BBN, in some cases the grammars used for these tests have been adapted to this finite set of sentences, including the test material.

VOCABULARY/LEXICON/OUTPUT CONVENTIONS

The task domain sentences in effect define the vocabulary. Internal representations (lexicon entries) may be at the system designer's choice, but for the purposes of implementing uniform scoring procedures, a convention was defined, drawing on material provided by CMU [5], BBN and TI. This convention includes the following considerations:

Case differences are not preserved. All input (reference) strings and output strings are in upper case.

There is no end-of-sentence punctuation. Nor is there any required special symbol to denote silences (either pre-pended, within the sentence utterance, or appended) or to indicate failure of a system to parse the reference string or input speech.

Apostrophes are represented by plusses. Words with apostrophes (embedded or appended) are represented as single words. Thus "it's" becomes "IT+S".

Abbreviations become single words. All periods indicating abbreviations are removed and the word is closed up (e.g. "U. S. A." becomes "USA").

Hyphenated items count as single words. In general, compound words that do not normally appear as separate words in the context of the assumed task domain model are entered as single, hyphenated items. The exception to this rule are compounds that include a geographic term, such as STRAIT, SEA or GULF. Thus entries such as the following count as single "words": HONG-KONG, SAN-DIEGO, ICE-NINE, PAC-ALERT, LAT-LON, PUGET-1, M-RATING, C-CODE, SQQ-23, etc. However, BERING STRAIT is to count as two words since this compound includes the geographic term "STRAIT", and it is not to be hyphenated.

Acronyms count as single words, and the output representation is not the form of the acronym made easier to interpret or pronounce (e.g. "PACFLT", not PAC-FLEET or PAC FLEET).

Mixed strings of alpha-numerics are treated as acronyms. Thus, "A42128" is treated as a one-word acronym, even though the prompt form of this indicates that this is to be pronounced as "A-4-2-1-2-8". Strings of the alpha set are also treated as acronyms (e.g. "USA"). Strings of digits are entered in a manner that takes into account the context in which they appear. Thus for a date such as 1987, it is represented as three words: "NINETEEN" "EIGHTY" "SEVEN". If it is referred to as a cardinal number it would be represented as "ONE" "THOUSAND" "NINE" "HUNDRED" "EIGHTY" "SEVEN".

SCORING THE TEST MATERIAL

For results to be reported at the March meeting, the use of different scoring software will be acceptable. Each contracter was free to use software consistent with the following general requirements:

Data are to be reported at two levels: sentence level and word level.

At the sentence level, a sentence is to be reported as correctly recognized only if all words are correctly recognized and there are no deletion or insertion errors (other than insertions of a word or symbol for silence or a pause). The percent of sentences correctly recognized is to be reported, along with the percent of sentences that contain (at least one) insertion error(s), the percent of sentences that contain (at least one) deletion error(s) and the percent of sentences that contain (at least one) substitution error(s). The number to be used for the denominator in computing these percentages is the number of input sentences in the relevant test subset, without allowing for rejection of sentences or utterances that may not parse or for which poor scores result.

At the word level, data that are to be reported include the percent of words in the reference string that have been correctly recognized. For these tests, "correct recognition" does not require that any criterion be satisfied with regard to word beginning or ending times. It is valuable, but not required, to report the percent of insertion, deletion, and substitution errors occurring in the system output.

For those systems that provide sentence or word lattice output, scoring should be based on the top-ranked sentence hypothesis. Additional passes through the alternative hypotheses are acceptable, provided the data are compared with comparable data for the top-ranked hypothesis.

System response timing statistics should be reported.

Data resulting from these tests is to be provided to NBS following the March meeting for detailed analysis and in evaluating alternative scoring software.

DOCUMENTATION

Documentation on the characteristics of the imposed grammar(s) must be provided. This information should describe any use of the material from which the test material was drawn (i.e. the set of 2200 task domain sentences developed at BBN and used by TI in recording the Resource Management Speech Database).

The system architecture and hardware configuration used for these tests should be documented.

REFERENCES

[1] D.S. Pallett, "Selected Test Material for the March 1987 DARPA Benchmark Tests",

Proceedings of the March 1987 DARPA Speech Recognition Workshop.

[2] (anonymous) "Integration, Transition and Performance Evaluation of Generic Artificial Intelligence Technology", Strategic Computing Program draft document dated Dec.6, 1985 (For Official Use Only).

[3] D.S. Pallett, "Benchmark Test Procedures for Continuous Speech Recognition Systems", draft document dated August 29, 1986 distributed prior to the Fall 1986 DARPA meeting.

[4] D.S. Pallett, "Performance Assessment of Automatic Speech Recognizers", Journal of Research of the National Bureau of Standards, Volume 90, Number 5, September-October 1985, pp. 371-387.

[5] A.I. Rudnicky, "Rules for Creating Lexicon Entries", note dated 11 February, 1987.

SELECTED TEST MATERIAL

FOR THE

MARCH 1987 DARPA BENCHMARK TESTS

David S. Pallett

Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, MD 20899

ABSTRACT

This paper describes considerations in selecting test material for the March '87 DARPA Benchmark Tests. Using a subset of material available from the Task Domain (Resource Management) Development Test Set, two sets of 100 sentence utterances were identified. For Speaker Independent technology, 10 speakers each provide 10 test sentences. For Speaker Dependent technology, 4 speakers each provide 25 test sentences. For "live talker" test purposes, three 30-sentence scripts were identified, using a total of 70 unique sentence texts. The texts of all of these test sentences were drawn from a set of 2200 sentences developed by BBN in modelling the (resource management) task domain.

INTRODUCTION

In order to implement benchmark tests of speech recognition systems to be reported at the March '87 DARPA Speech Recognition Meeting, it was necessary to specify selected test material. This test material is drawn from two sources: (a) the Task Domain Speech Database recorded at Texas Instruments (also referred to as the "Resource Management" Database), and (b) the use of "live talkers" in site visits. In each case, the texts of the sentences were drawn from a set of sentences developed by BBN. Selection of test material using the Resource Management Database includes two separate components, a Speaker Independent component and a Speaker Dependent component. This paper outlines the process of defining these subsets of speech material.

At the time the Resource Management Speech Database was designed, it was intended that approximately equal volumes of material would be available for system development (research) purposes and for two rounds of benchmark tests. Consequently, approximately half of the available material is designated "development" or "training" material, and the remaining portion is designated for test purposes. The test material is designated as "Development Test" or "Evaluation Test" sets, each including 1200 test sentence utterances in each portion (Speaker Independent or Speaker Dependent).

The design and collection of this Task Domain (Resource Management) Speech Database is described elsewhere in this Proceedings in a paper by Fisher [1].

Thus, as originally intended, two sets of 1200 sentence utterances were to be available for the March '87 tests. During January 1987, discussions involving representatives of CMU, BBN, MIT, NBS and the DARPA Program Manager determined that use of this large a volume of test material was not necessary to establish performance of current technology when pragmatic considerations of processing times and expected performance levels were made. Consequently, it was agreed that subsets of 100 sentence utterances were to be defined for these tests, and that NBS would specify the appropriate subset.

To complement the use of the recorded speech database material, a test protocol for the use of "live talkers" emulating in some sense procedures to be used in future demonstrations of these systems was defined, and texts were selected for this purpose.

RESOURCE MANAGEMENT SPEECH DATABASE TEST MATERIAL

Speaker Independent Test Material

For the March'87 tests, a set of ten speakers was identified, drawn from material recorded at TI and made available to NBS in December '86 and January '87

Each speaker provided two "dialect" and the ten "rapid adaptation" sentences in addition to a total of thirty test sentence utterances. For each speaker, a unique subset of ten sentence utterances were specified to be used for the March '87 tests, amounting to 100 sentence utterances in all (10 speakers times 10 sentence utterances per speaker).

Seven male speakers were selected and three female speakers, reflecting the male/female balance throughout the Resource Management Speech Database.

To aid in the selection of individual speakers, a set of approximately 16 speakers was identified. SRI was asked for advice on whether any of these would be regarded as anomalous on the basis of the "dialect" sentences obtained in the acoustic-phonetic database. SRI performed a clustering analysis and advised us that most of the speakers clustered in three groups of similar speakers with three other individuals categorized as exceptional in some sense (e.g. unusually slow rate of speech) [2]. The ten speakers identified for inclusion in the test subset include one of these "exceptional" speakers, the others being drawn from the three clusters to provide some degree of coverage of regional effects.

Table 1 provides detailed information on the individual speakers' regional backgrounds, race, year of birth and educational level for the ten selected speakers in the March '87 Test Subset.

Analysis, by TI, of the lexical coverage provided by this subset of the test material indicates that 348 words occur at least once in this test material, and the total number of words is 836, for a mean length of each sentence of 8.36 words.

Subject	Sex	Region	Race	Year of Birth	Education
DAB	MALE	NEW ENGLAND	WHT	'62	B.S.
GWT	MALE	NORTHERN	WHT	'21	B.S.
DLG	MALE	NORTH MIDLAND	WHT	'42	(?)
CTT	MALE	SOUTHERN	WHT	'62	B.S.
JFC	MALE	NEW YORK CITY	WHT	'59	B.S.
BTH	MALE	WESTERN	WHT	'62	B.S.
[illegible]	FEMALE	SOUTHERN	WHT	'58	B.S.
BCG	FEMALE	"ARMY BRAT"	(?)	'59	B.S.
DAH	FEMALE	NEW ENGLAND	WHT	'46	B.S.
JFP	MALE	WESTERN	WHT	'39	M.S.

Table 1. Speaker Independent Test Subset

Speaker Dependent Test Material

For these tests, a set of four speakers was identified, also drawn from material recorded at TI and made available to NBS during December '86 and January '87. In this case, selection of the specific individuals was strongly influenced by the availability of training material. BBN expressed concern that the entire set of 600 sentence utterances intended for system training should be available for any test speakers. At the time of selection of test material, not all of the 12 speakers for this portion of the database had completed recording their training material. With this in mind four speakers were identified.

Each speaker had previously recorded the ten "rapid adaptation" and "dialect" sentences, and the Development Test material included 100 sentence utterances for each speaker. From this, unique sets of 25 sentence utterances were identified for each of the four speakers, amounting to 100 sentence utterances in all for this portion of the test material.

Three of the speakers were male and one was female.

Table 2 provides additional data on these speakers.

Analysis, by TI, of the lexical coverage provided by this subset of the test material indicates that 832 words occur at least once, with a total number of words of 832, for a mean sentence length of 8.32 words. This is quite similar to that for the Speaker Independent material, although the details of the distributions differ slightly.

Subject	Sex	Region	Race	Year of Birth	Education
CMR:	FEMALE	NORTHERN	WHT	'51	M.S.
BFF:	MALE	NORTH MIDLAND	WHT	'52	Ph.D
JWS:	MALE	SOUTH MIDLAND	WHT	'40	B.S.
RKM:	MALE	SOUTHERN	BLK	'56	B.S.

Table 2. Speaker Dependent Test Subset

LIVE TALKER TEST MATERIAL

For the "live tests", it was necessary to select sentence texts that would be read by the test speakers. It was thought desirable to use three speakers, each speaker reading a total of 30 sentence texts in addition to the 10 "rapid adaptation" sentences. Ten of the thirty sentence texts were to be the same

for all speakers, so that of the 90 sentence utterances to be used for testing, there would be three productions of each of the ten sentences, and 60 other sentences (20 for each of three speakers). A total of 70 unique sentence texts was thus required.

The sentence texts were selected frc a subset of 2200 Resource Management sentences. CMU representatives had indicated a preferance for sentence texts that could be produced in less than 6 seconds. Accordingly, the essentially random process of sentence text selection was perturbed slightly to throw out longer sentences.

Lexical analysis, by TI, of the scripts developed from these sentences indicates that the three scripts are well-balanced in terms of mean sentence length and number of lexical entries. Each of the three scripts has a mean sentence length of 7.93 words (258 words/30 sentences), reflecting the intentional bias in sentence selection process toward slightly shorter sentences. The number of lexical entries in the three scripts is 153, 155 and 161.

The prompt form of each of these scripts was to be made available to the "live talkers" in site visits to be conducted in March '87. Each of the test speakers was to use the Sennheiser HMD 414-6, the same microphone used at TI for the Resource Management Speech Database, and the test environment was to be a computer lab or conference room with no competing conversation. A portion of the test material was to be provided in an interactive manner (i.e. while waiting for system processing of the data) and the remainder was to be processed off line.

GRAMMATICAL COVERAGE

At the time that BBN developed the set of approximately 2800 sentence texts modelling this task domain, no explicit or formally defined grammar was used. Rather, a set of prototypical sentences was identified to provide coverage of the task, and the subset of vocabulary occurring in these sentence "patterns" was then expanded to approximately 1000 words. There were a total of approximately 950 sentence patterns [3]. By incorporation of the expanded vocabulary, the 2800 sentences were generated by including approximately three exemplars of each pattern. From these, 600 were designated to be used for speaker-dependent training material, leaving a remaining subset of 2200 sentences. All of the test material was randomly selected from this subset of 2200 sentences.

No analysis to determine the representation of the basic sentence patterns in the test material has been conducted to date.

REFERENCES

[1] W. A. Fisher, "A Task Domain Database", Proceedings of the March 1987 DARPA Speech Recognition Workshop.

[2] J. Bernstein, private communication, January 1987.

[3] P. Price et al., oral presentation at the September 1986 DARPA Speech Recognition Workshop.

ROBUST HMM-BASED SPEECH RECOGNITION: AN UPDATE

Clifford J. Weinstein

Lincoln Laboratory, Massachusetts Institute of Technology
Lexington, Massachusetts 02173-0073

INTRODUCTION

Lincoln Laboratory work in robust speech recognition through February 1986 is summarized in the Proceedings of the prior DARPA Speech Recognition Workshop [1]. The papers included in these proceedings [2-5] provide an update on major technical accomplishments over the past year. This summary provides an overall introduction to the accompanying papers, and notes some current efforts and some additional accomplishments of the Lincoln program in robust speech recognition.

OVERVIEW OF TECHNICAL APPROACH TO ROBUST RECOGNITION

Our approach to achieving high-performance recognition of speech produced under stress and in noise has been to develop techniques for enhancing the robustness of a baseline Hidden Markov Model (HMM) recognizer. The training and recognition modules of a baseline isolated-word HMM system are depicted in Fig. 1, while Fig. 2 indicates the robustness enhancements which have been developed and tested. Descriptions of the various enhancements and their effectiveness are described in the accompanying papers. Many of the enhancements, such as grand variance or temporal difference parameters, are in the area of improved modelling and training in the framework of the basic HMM system. Other enhancements, such as the second-stage discriminant analysis system, are outside the basic HMM framework.

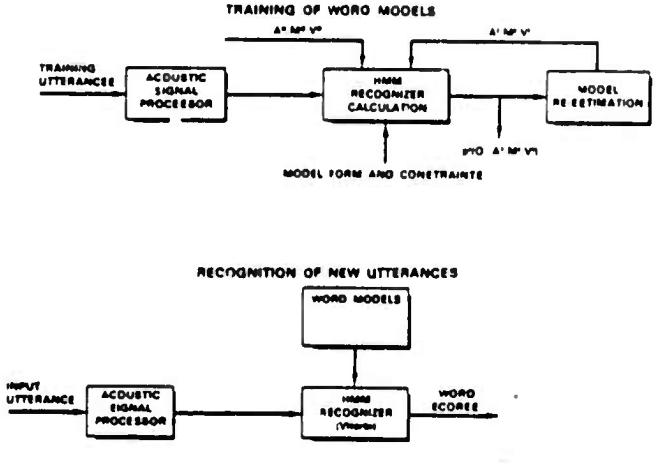


Fig. 1. Hidden Markov Model isolated-word recognition system.

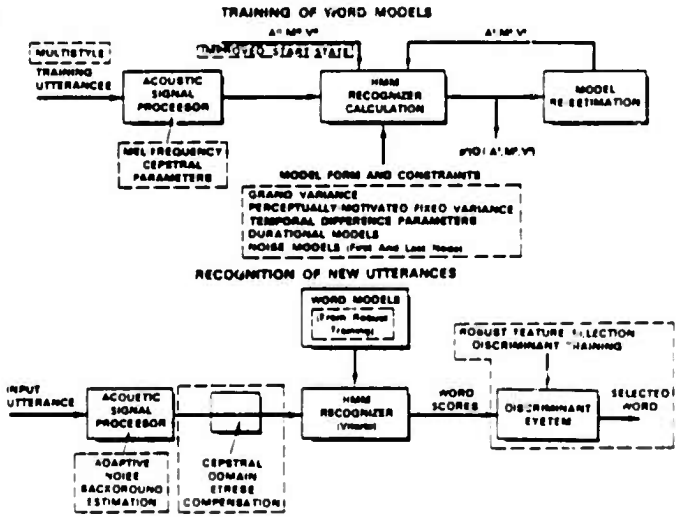


Fig. 2. HMM isolated-word recognition system with robustness enhancement.

DATA BASES OF SPEECH PRODUCED UNDER STRESS AND IN NOISE

Two primary data bases have been used for the isolated-word robust recognition algorithm development work: (1) the "TI-stress" 105-word vocabulary data base [6], including simulated-stress through talker style variation and noise exposure (Lombard condition); and (2) the "Lincoln-stress" data base [2], including simulated-stress via talker-style variation, Lombard condition, and workload stress, with a 35-word vocabulary composed of acoustically-similar subsets of the TI 105-word vocabulary. Additional experiments have been conducted on "TI-20," a standard, normally spoken 20-word vocabulary data base [7].

HIGHLIGHTS OF ACCOMPANYING PAPERS [2-5]

The basic robust HMM isolated-word recognizer, and experiments and results on TI-stress and TI-IWD, are described in [2]. A variety of robustness techniques are described, and the results on the effectiveness of various techniques are compared and discussed. Recognition accuracy results reported for TI stress are 98.05% with training restricted to normal speech only, and 99.12% with training on multiple speech styles. These represent more than an order-of-magnitude improvement relative to a baseline HMM, as well as significant improvement relative to results reported in [1]. In addition, the best results known to date for any system are reported for TI-IWD: 99.94%, first test; 100%, best test. This shows that the robustness techniques apply effectively to normal speech variations, as well as to the stress variations for which they were developed.

The focus of [3] is a particular robustness enhancement technique wherein the basic recognition parameters (mel-frequency cepstra) are modified adaptively to compensate for variations due to stress. This adaptation is shown to compensate for spectral tilt and to produce significant performance improvements for systems trained with normal speech.

Multistyle training, and experiments and results on the Lincoln-stress data base, are the focus of [4]. The effectiveness of training on multiple talker styles in improving recognition performance for stress and noise conditions (workload, Lombard) not included in the training data is reported and discussed. Overall recognition accuracy of 99% on the difficult Lincoln-stress data is reported, achieved via a combination of multi-style training and other robustness enhancements.

A second-stage discriminant analysis system, developed to serve as a post-processor to the HMM recognizer, in order to resolve confusion between acoustically-similar words, is described in [5]. This discriminant system is trained by passing samples of every word in the vocabulary through the HMM models of every word in the vocabulary, to explicitly model acoustic differences between words. A statistically-based sifting technique is described which selects only those parameters which are likely to be effective in discrimination. Performance improvements relative to the robust single-stage HMM are reported for the Lincoln-stress data base, contributing, for example, to the overall 99% (see above) recognition accuracy on that data base.

CURRENT EFFORTS

A number of current efforts in progress, and recent accomplishments not covered in the accompanying papers, are outlined here. More will be reported in this work in the future.

A major focus of our current efforts is to extend the robust recognition work to operate with high performance on continuously-spoken sequences of words, under conditions of stress and noise. Our work is directed specifically at limited-vocabulary, restricted task domains representative of the Pilot's Associate application of the DARPA Strategic Computing Program. To this end, a prototype continuous speech HMM trainer and recognizer has been brought into operation and subjected to initial testing. The system trains on continuous speech, can use either subword models or whole-word models, and includes relevant robustness techniques used in the isolated-word system. The continuous-speech recognition (CSR) system is operating with good preliminary results. The new CSR system has also been interfaced to our live-input front end (see [2]) and used effectively in numerous demonstrations.

Another important focus of our current work is adaptation of the robust recognizer to the environment and to the talker, by modifying the parameters of the recognizer (e.g., the HMM word models) during operation. Adaptation work has been conducted so far in the context of isolated-word recognition. Encouraging preliminary results have been obtained both through adaptation of the basic HMM system and through adaptation of the second-stage discriminator. A variety of techniques are being developed and compared, ranging from full retraining of the HMM model to simple adaptation of the cepstral means.

Finally, since one of our target goals is recognition in the aircraft cockpit, we are developing a simulated aircraft scenario for a demonstration system. The goal is a realistic, stressful flight task for voice control on an aircraft. Currently, a prototype flight simulator has been developed on a SUN workstation. The simulator provides approximate models for three aircraft types--a Cessna 150, an F-15, and a high-altitude powered glider. A number of improvements to the simulator (e.g., improved weather model and navigation aids) are planned for the near future; later, the simulator will be interfaced to the speech recognizer, and a suitable control language will be designed and implemented.

REFERENCES

[1] D. B. Paul, R. P. Lippmann, Y. Chen, C.J. Weinstein, "Robust HMM-Based Techniques for Recognition of Speech Produced Under Stress and in Noise," Proceedings DARPA Speech Recognition, February 1986; also published in Speech Tech 86 Conference Proceedings, April 1986.

[2] D. B. Paul, "A Speaker-Stress Resistant HMM Isolated Word Recognizer," these proceedings; also published in Proceedings ICASSP 87, April 1987.

[3] Y. Chen, "Cepstral Domain Stress Compensation for Robust Speech Recognition," these proceedings; also published in Proceedings ICASSP 87, April 1987.

[4] R.P. Lippmann, E.A. Martin, and D.B. Paul, "Multi-Style Training for Robust Speech Recognition," these proceedings; also published in Proceedings ICASSP 87, April 1987.

[5] E. A. Martin, R. P. Lippmann, and D. B. Paul, "Two-Stage Discriminant Analysis for Improved Isolated-Word Recognition," these proceedings; also published in Proceedings ICASSP 87, April 1987.

[6] P.J. Rajasekaran, G.R. Doddington, and J.W. Picone, "Recognition of Speech Under Stress and in Noise," Proceedings ICASSP 86, April 1986.

[7] G. R. Doddington and T. B. Schalk, "Speech Recognition: Turning Theory into Practice," IEEE Spectrum, September 1981.

A SPEAKER-STRESS RESISTANT HMM ISOLATED WORD RECOGNIZER

Douglas B. Paul

Lincoln Laboratory, Massachusetts Institute of Technology
Lexington, Massachusetts 02173-0073

ABSTRACT

Most current speech recognition systems are sensitive to variations in speaker style. The following is the result of an effort to make a Hidden Markov Model (HMM) Isolated Word Recognizer (IWR) tolerant to such speech changes caused by speaker stress. More than an order-of-magnitude reduction of the error rate was achieved for a 105-word simulated-stress data base and a 0% error rate was achieved for the TI 20 isolated-word data base.

INTRODUCTION

Current recognition algorithms are generally far more sensitive to variations in speaking style and conditions than are human listeners. Many factors can cause such changes in speaking style in an operational environment. For instance, typical causes are time (a week or more), nasal congestion, emotional state, expression, and task-induced stress. We are specifically interested in task-induced stress, but the acoustic changes appear to be similar for many causes. Typical effects of stress are changes in spectral tilt, formant position, energy, timing, and phonetic content. The following describes work which has yielded more than an order-of-magnitude reduction in the error rate of an HMM IWR over a multi-speech-style data base as well as significant improvements for normally spoken speech.

Since it is difficult to obtain large amounts of data from stressed subjects, we have used a multi-style simulated-stress data base generated at TI [1]. This data base has 8 speakers (5M + 3F), a 105-word aircraft vocabulary and, for each speaker, a (normally spoken) 5 token per word training section and 6 style sections of 2 tokens per word each for testing. The speech was digitized with a 4 kHz audio bandwidth. The six conditions are: normal, fast, loud, Lombard (noise presented in headphones), soft, and shout. The shout condition is so different from the other conditions that it has been largely ignored. The work has focused on the other 5 conditions with their overall average substitution error rate (avg5) as the primary measure of performance.

The observations used by this system are centisecond mel-cepstra [2]. These mel-cepstra are computed from the digitized speech by the following processing sequence:

1. 20 msec Hamming window.
2. 256 point FFT (⟶ complex spectrum).
3. Magnitude squared (⟶ power spectrum).
4. Preemphasis: $S(f) = S(f) \cdot (1+(f/500\text{Hz})^2)$.
5. Triangular mel-bandpass summations (⟶ mel power spectrum). Constant area filters: 100 Hz spacing between .1 and 1 kHz, 10% above; width = 2x spacing.
6. Convert mel power spectrum to dB.
7. Modified cosine transform [2] (⟶ mel-cepstrum).

Similar mel-cepstral observations have been used successfully in a number of recognition systems. See, for example, [3].

The basic system is a diagonal-covariance-matrix multi-variate Gaussian probability density continuous-observation [4] isolated-word HMM recognizer using the above mel-cepstral observations. (The absolute energy term, c0, is not used. The systems augmented with differential observations (see below) include a differential energy term.) Only one model is used per word. The system is trained by the Baum-Welch (forward-backward) algorithm. Since the data files contain a few hundred ms of background noise at each end, the first and last nodes of the model are dedicated to modeling the background. (Both the training and recognition are open endpoint). The termination of the observations is modeled by a transition to a degenerate node. The recognizer uses a Viterbi decoder. All systems reported here use "linear" networks--i.e., there are no model skip transitions.

A variety of training conditions and HMM systems were tested. The training conditions are "normal" training, where only the normally spoken training section of the data base was used for training, and "multi-style" training where the first token of each word of each style was added to the training set and the second token was used for testing. Variations in the systems are of several forms: number of nodes per word, observation enhancements, the method of obtaining the variances, the training start state, use of adaptive background nodes during recognition, the duration model, and modifications to improve parameter estimates in the face of small amounts of training data. Unless otherwise stated, all systems have 10 active nodes.

NORMAL TRAINING

The normal training systems used 5 normally spoken tokens per word for (speaker-specific) training and a total of 1680 tokens per style (8400 per avg5) for testing. Each system will be identified by a code. The error rates quoted are the avg5 percent error. More detail will be found in Table 1 and Figure 1.

The baseline system used trained model variances (baseline, 1%: 20.49%). Lower bounding the variances (variance limiting), which reduced the effects of limited

training data, reduced the error rate to (v1,10: 15.92%). Augmenting the observations with 20 ms temporal differences of the mel-cepstra further reduced the error rate to (v1,d2,10: 10.50%).

The above systems all used individually trained variance vectors for each node. The following systems use the same variance vector for all nodes. An L2 norm (all variances = 1) performed poorly (L2, 10: 13.58%). Fixed variances equal to the variances of all the training speech of all speakers (Fig. 2) yielded (raw, 10: 8.76%). A perceptually-based fixed variance (see below and Fig. 2) reduced the error rate to (gfv,10: 6.13%). Adding 20 ms temporal differential parameters (gfv, d2,10: 4.99%), increasing the number of nodes to 14 (gfv, d2,14: 2.54%), modifying the fixed variance yielded (gafv,d2,14: 2.26%), and finally, adding an endpoint-based training start state and adaptive background estimation to the recognizer reduced the error rate to (gafv,d2,b,14: 1.95%). A trained "grand variance" (Fig. 2) computed in the Baum-Welch reestimation procedure, in which the variance is tied over all nodes of all words, approached the fixed variance (grandv, d2,b,14: 2.95%).

MULTI-STYLE TRAINING

The multi-style trained systems added 1 token from each non-shout test style to the training set for 10 tokens per word (mtns). (Similar results were obtained even when a shout token was included in the training.) This left 840 test tokens per style and 4200 tokens for the avg5. None of the test tokens were used for training.

In general, the results improved significantly:

normally trained		multi-style trained	
v1,d2,b,14:	7.71%	mtns,v1,d2,b,14:	1.12%
gafv,d2,b,14:	1.95%	mtns,gafv,d2,b,14:	.93%
grandv, d2,b,14:	2.95%	mtns,grandv,d2,b,14:	.88%

Multi-style training, by presenting more legitimate variation to the training algorithm than does the normal training, appears to cause the training to find a better model for the word [5]. In addition, performance on the normal style usually improves as a result of the multi-style training.

THE PERCEPTUALLY MOTIVATED FIXED VARIANCE

Vowel perception experiments [6] have indicated that formant position is more important than spectral tilt in vowel identification. Our investigations also showed large spectral tilts to be one of the effects of speaker stress and style [7]. However, spectral tilt is also a strong cue for distinguishing between voiced and unvoiced speech. Therefore, a weighting (inverse variance) which deemphasized, but did not totally eliminate the low-order mel-cepstral terms, was applied in the distance measure (i.e., was used to postulate a fixed variance in the Gaussian probability density function). The weighting function was chosen to be (Fig. 3):

$$w[i]=\begin{cases}\left(1-.8e^{-\left(\frac{i}{8}\right)^2}\right)^2 & 1<i<d \text{ (normal mel cepstra)}\\ 2\left(1-.8e^{-\left(\frac{i}{8}\right)^2}\right)^2 & 0<i<d \text{ (differential mel cepstra)}\\ 0 & \text{otherwise}\end{cases}$$

$$var[i] = 1/w[i]$$

where i is the mel-cepstral index and d and dd are the observation orders.

This weighting, if interpreted as a signal processing operation on the mel-spectrum, is a dynamic range compressor and local feature enhancer very similar to the homomorphic dynamic range compression and contrast enhancement techniques used in picture processing [8]. It also provides a degree of tolerance to changes in the audio channel (see below).

STRONGER DURATION MODELS

The Ferguson full duration model [9], which models the duration as a vector of duration probabilities rather than the dying exponential of the standard HMM, yielded mixed results when applied to some of the above systems. It is much more computationally intensive than the standard systems and has not been adequately explored. It also appears to require more training data than was available for these experiments.

A double-node subnet is much simpler and computationally more efficient than the full duration model. Each node is replaced with a network consisting of 2 series-connected nodes constrained to have the same observation probability density functions and the same self-transition probabilities. With no increase in the number of parameters and only a minor increase the total computation (the computational load is dominated by the probability density functions), the duration model is changed from the usual exp(-at) to a t*exp(-at) form. This system has not been adequately explored, but the results are encouraging: from (gafv,d2,10: 3.69%) to (dn,gafv,d2,10: 3.42%). Similar duration models have been examined elsewhere [10].

ADDITIONAL RESULTS

This system, which was developed using the TI simulated-stress data base, has been tested on two other data bases and in numerous live-input demonstrations. The system has shown similar results on a locally-generated, simulated, and workload stress data base [5,11]. The system has also been tested using the TI 20 isolated-word data base (16 speakers, 20-word vocabulary) [12]. The error rates were as follows:

First test:

ti-iwd: gfv,d2,10: [illegible] (3/5120)

Best result:

ti-iwd: gfvx,d2: .00% (0/5120)

The best result reported in [12] is .20% (10/5120).

Llve-lnput teste of the normally trained, fixed-variance eyetem have confirmed the robustness to epeech etyle and have lndicated additional tolerances. Thie eyetem has been trelned over a local dlaled-up telephone line and tested over dlaled-up long-dletence telephone llnee. A teet was performed when the epeaker had a cold. (The trelning data had been recorded monthe earller.) Under both of theee conditione, the eyetem has contlnued to perform well. Wlth the exception of the adaptive background nodes, the eyetem dld not adapt in any way to th~ teet environment.

The eyetem has been formally tested or demonetrated over bandwidthe ranging from 3.2 kHz to 8 kHz. (In each caee the testing and training bandwldthe were identlcal.) A variety of microphonee, audio eyetems, and background noiee levele have been used and eyetem has tolerated them, all wlthin reeeonable llmlte.

DISCUSSION

The lmprovemente reported here are the reault of eeveral phllooophies: the model muet be trelnable, muet have eufflclently detalled obeervatione, and muet be tolerant of unanticlpated changee. Any parameters used muet be trainable on realietically avallable emounte of data. Thue, the verlance limiting, grand verlance, and flxed-variance eyetems outperformed the baeeline eyetem. Augmentlng the obeervatione wlth time-differentlal parameters provlded more lnformatlon to the eyetems and thue yielded further lmprovemente. Hidden Markov modele are fairly lneenelttive to the exact number of nodes, but the lncreeee from 10 to 14 nodes generally improved their ability to model the given vocebulary.

A "fully trelned" model can only model the trelning data--lt cannot actively entlclpate what lt has not seen. We provide a priori lnformatlon to theee eyetems ln eeveral waye. The number of nodes and the allowable transitions between them are one form of such a priori lnformation. The fixed variance le another way of provlding uaeful a priori knowledge. The normally trelned grand variance eyetem only knowe about epeech veriatlone found in lte trelning data. The flxed variance lnforme the eyetem about the klnde of verlatlone which may be encountered ln epeech and, therefore, outperforme the grand verlance for normal trainlng wlth etyle teetlng, and gives equivalent performance for normal trelning wlth normal teetlng and multi-etyle trelning wlth etyle teetlng. From another vlewpoint, the application of a priori knowledge has reduced the requirements for trelning data by anticipating the variation ln the teet data.

The techniques deecribed here lncreeee the tolerance of the recognizer to epeech veriatlone. Another approach ueed a flxed verlance, normally trelned eyetem, modified to compenaate for the epectral tilte during recognition [13].

CONCLUSIONS

Several techniquee for improving the training and speech modeling of a "textbook" (baeeline) HMM recognizer with good normal epeech performance have been combined to significantly improve recognition reeults in the face of speech-etyle variation and emall amounts of training data. Reeults have improved from a 20.5% avg5 error rate to 1.95% if normal training data ie ueed, or to .88% if samplee of the expected speech etyles are available. Theee enhancemente have alao improved performance on normal epeech, ae ehown by the .24% error rate echieved for normal epeech, and the .06% and .00% error rates achieved on the TI 20 lsolated-word data baee.

ACKNOWLEDGEMENT

Thie work was done in conjunction with my colleaguee, Y. Chen, E.A. Martin, and R.P. Lippmann.

REFERENCES

[1] P.K. Rajasekaran, G.R. Doddington, and J.W. Picone, "Recognition of Speech Under Stress and in Noise," ICASSP '86, Tokyo (April 1986).

[2] S.B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," IEEE Trans. ASSP (August 1980).

[3] O. Jouvet and R. Schwartz, "One-Pass Syntax-Directed Connected-Word Recognition in a Time-Sharing Environment," ICASSP '84, San Diego (March 1984).

[4] L.R. Bahl, R. Bakis, P.S. Cohen, A. Cole, F. Jelinek, B.L. Lewis, and R.L. Mercer, "Continuous Parameter Acoustic Processing for Recognition of a Natural Speech Corpus," ICASSP '81, Atlanta (April 1981).

[5] R.P. Lippmann, E.A. Martin, and D.B. Paul, "Multi-Style Training for Robust Isolated-Word Speech Recognition," ICASSP '87, Dallas (April 1987).

[6] R. Carlson, B. Granstrom, and D. Klatt, "Vowel Perception: The Relative Perceptual Salience of Selected Acoustic Manipulations," Speech Communication Papers Presented at the 97th Meeting of the ASA, Boston (June 1979).

[7] D.B. Paul, R.P. Lippmann, Y. Chen, and C.J. Weinstein, "Robust HMM-Based Techniques for Recogniton of Speech Produced Under Stress in Noise," Speech Tech '86, New York (April 1986).

[8] B. Gold and C.M. Rader, "Digital Processing of Signals," McGraw-Hill, New York (1969).

[9] J.D. Ferguson, "Variable Duration Models for Speech," Proc. Symp. on Application of Hidden Markov Models to Text and Speech, J.D. Ferguson, Ed., Princeton, NJ, pp. 143-179 (1980).

[10] A.E. Cook and M.J. Russell, "Improved Duration Modeling in Hidden Markov Models Using Series-Parallel Configurations of States," Proc. Institute Acoust. Conf. on Speech and Hearing (1986).

[11] E.A. Martin, R.P. Lippmann, and D.B. Paul, "Two Stage Discriminant Analysis for Improved Isolated-Word Recognition," ICASSP '87, Dallas (April 1987).

[12] G.R. Doddington and T. B. Schalk, "Speech Recognition: Turning Theory into Practice," IEEE Spectrum (September 1981).

[13] Y. Chen, "Cepstral Domain Stress Compensation for Robust Speech Recognition," ICASSP '87, Dallas (April 1987).

TABLE I
SUBSTITUTION ERRORS FOR TI-SIMULATED STRESS AND TI-IWD DATA BASES

	normal	avg5
Normal Training:		
individual nodal variances:		
baseline	1.90	20.49
v1,10	1.07	15.92
v1,d2,10	.65	10.50
v1,d2,14	.48	8.57
v1,d2,b,14	.48	7.71
same fixed variance for all nodes:		
L2,10	2.08	13.58
raw,10	.95	8.76
gfv,10	.65	6.13
gfv,d2,10	.36	4.99
gfv,d2,14	.36	2.54
gafv,d2,14	.48	2.26
gafv,d2,b,14	.36	1.95
same trained variance for all nodes:		
grandv,d2,b,14	.36	2.95
Multi-Style Training:		
individual nodal variances:		
mtss,v1,d2,b,14	.71	1.12
same fixed variances for all nodes:		
mtss,gafv,d2,b,14	.48	.93
same trained variance for all nodes:		
mtss,grandv,d2,b,14	.24	.88
TI-IWD (20 word)		
gfv,d2		
first test: .06		
best: .00		

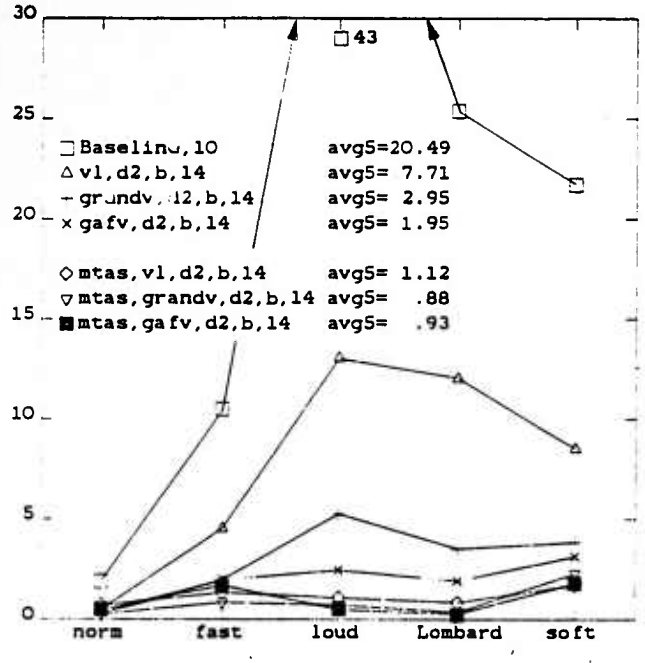


Fig. 1. Percent substitution errors for TI simulated stress data base. See text for system codes.

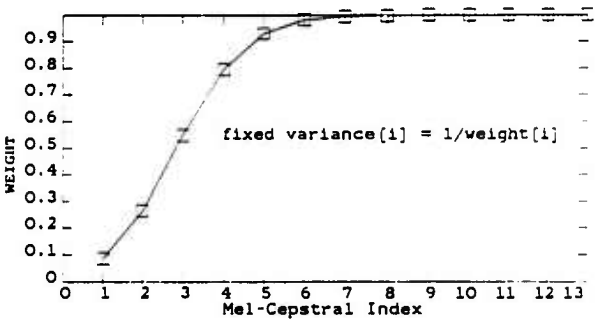


Fig. 3. The perceptually-motivated weighting.

This paper appears in the Proceedings of ICASSP 87.

This work was sponsored by the Defense Advanced Research Projects Agency.

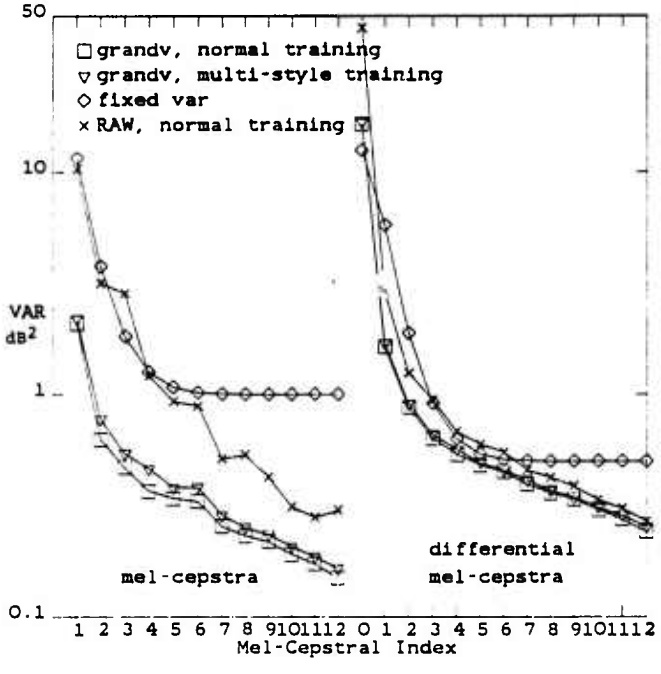


Fig. 2. Variances for several training techniques.

CEPSTRAL DOMAIN STRESS COMPENSATION FOR ROBUST SPEECH RECOGNITION

Yeunung Chen

Lincoln Laboratory, Massachusetts Institute of Technology
Lexington, Massachusetts 02173-0073

ABSTRACT

Automatic speech recognition algorithms generally rely on the assumption that for the distance measure used, intraword variabilities are smaller than interword variabilities so that appropriate separation in the measurement space is possible. As evidenced by degradation of recognition performance, the validity of such an assumption decreases from simple tasks to complex tasks, from cooperative talkers to casual talkers, and from laboratory talking environments to practical talking environments.

This paper presents a study of talker-stress-induced intraword variability, and an algorithm that compensates for the systematic changes observed. The study is based on Hidden Markov Models trained by speech tokens in various talking styles. The talking styles include normal speech, fast speech, loud speech, soft speech, and talking with noise injected through earphones; the styles are designed to simulate speech produced under real stressful conditions.

Cepstral coefficients are used as the parameters in the Hidden Markov Models. The stress compensation algorithm compensates for the variations in the cepstral coefficients in a hypothesis-driven manner. The functional form of the compensation is shown to correspond to the equalization of spectral tilts.

Preliminary experiments indicate that a substantial reduction in recognition error rate can be achieved with relatively little increase in computation and storage requirements.

INTRODUCTION

Current speech recognition systems generally degrade significantly in performance if the systems are not both trained and tested under similar talking conditions. A major reason for performance degradation when testing and training conditions differ is that people speak differently under different conditions. Despite the knowledge that speech patterns change in stress and in noise, little speech recognition research has been directed at modeling systematic changes observed and at developing recognition systems that are resistant to such changes.

This paper presents a study of talker-stress-induced variations in speech cepstral coefficients, and an algorithm that compensates for systematic (but unknown) changes observed. The study is based on isolated-word Hidden Markov Model speech recognizer [1] trained by speech spoken in various talking conditions. The recognizer [1] is a continuous-observation HMM system using mel-frequency cepstral parameters. The work reported in this paper is described in more detail in [2].

The experiments conducted in this research were based on the "simulated stress" [3] speech data base collected by Texas Instruments.

In this data base, stress-like degradations of the speech signal were elicited by asking the speaker to produce speech in a variety of styles (normal, fast, loud, soft, and shout) as well as with 95-dB pink noise exposure in the ear to produce the Lombard effect. The vocabulary consisted of 105 words, including monosyllabic, polysyllabic, and confusing words.

The data base was divided into training data and test data. Training data consisted of five samples of each of the 105 words collected in a random order under normal talking conditions, and test data consisted of two samples of each word under each simulated-stress condition. Data were collected from five adult males and three adult females. The total number of test word tokens was 10,080.

AN EXPERIMENT ON MULTISTYLE-TRAINED HIDDEN MARKOV WORD MODELS

Multistyle training [4,5] is a technique used to improve speech recognition performance under stress. In multistyle training a recognizer is trained using word tokens spoken with different talking styles instead of using words all spoken normally. It has been found to be easy for a talker to change to styles such as fast, slow, loud, and soft, producing changes in speech characteristics that are similar to changes that occur under stress.

An experiment on multistyle-trained Hidden Markov Model word recognition was performed. In this experiment, 11 speech tokens were used to train each word model: 5 tokens from the training data base, and 6 tokens, one per talking style except normal, from the test data base. The recognition error rates are listed in Table I. For comparison, the error rate of the baseline HMM system is also included.

From Table I we observe that there is dramatic performance degradation when the baseline recognizer is tested with styled data; and that multistyle-training has considerably improved system performance for speech data of all styles.

It appears that the HMM word models were able to assimilate the data from the multiple styles and to capture

statistically the more invariant features of each word. In the next section we investigate the gross changes of model parameters resulting from multistyle training as well as from style training (as opposed to normal training).

CEPSTRAL DOMAIN STRESS COMPENSATION-DRIVEN BY OBSERVATIONS

The success of the multistyle training experiment motivated a comparison of the model parameters trained under various talking styles to determine whether it would be possible to compensate for the cepstral changes through simple transformations on the cepstral means and variances obtained using normal training. Such transformation, if effective, would simplify the training procedure.

The differences among normally trained, single-style-trained, and multistyle-trained word models are partially reflected in the average shifts of the mean values and in the average scaling of the variances of the cepstral coefficients. To study each difference, seven different sets of word models were examined. Six of the models were trained under six individual conditions (normal, fast, loud, Lombard, soft, and shout respectively) while the seventh was trained using a composite of all these conditions (multi-style). The cepstral means and variances, averaged over all words in the T1 vocabulary, over all speech nodes in each word, and over all talkers, were computed for each of the models above.

The mean cepstral shifts (i.e., cepstral means of the given model minus the cepstral means of the normal model) for each of the cepstral coefficients are plotted in Fig. 1. Figure 1(a) plots mean cepstral shifts for four cases: soft; shout; average of fast, loud and Lombard; and multistyle. Figure 1(b) plots the corresponding spectra of these mean shifts, contrasting the effects of spectral tilt of low vocal effort (soft) vs higher vocal effort (fast, loud, Lombard, and shout). Increased vocal effort increases the relative high frequency content, whereas the opposite occurs with low vocal effort.

It is well known that spectral tilt exhibits large variation when a talker speaks under stress. Such variation usually contaminates the distance measure and is one of the most significant causes of recognition performance degradation. It appears that the effect of spectral tilt could be compensated, to some extent, by applying the appropriate cepstral compensation to normally trained word models.

Because variance estimation is less reliable than mean estimation, we have only compared cepstral variances of multistyle-trained models which used 11 training tokens with the normally trained models. Their ratios (multistyle/normal) are plotted in Fig. 1(c). It appears that the major style-induced variations occur in the most slowly varying spectral components (corresponding to lower order cepstral coefficients), and in most rapidly varying spectral components (corresponding to the higher order coefficients)[1].

The following cepstral compensation experiments were performed, in which new word models were generated by modifying normally trained Hidden Markov word models by one or more sets of cepstral differences. The word models were talker-dependent, but the modifications were the same for all words and all talkers.

(a) Single Model Compensation: The set of cepstral mean differences and variance ratios observed in multistyle-trained models [represented by filled squares in Fig. 1(a) and (c)] was applied as compensation in recognition tests on all styles.

(b) Multimodel Compensation: Three sets of cepstral mean compensations corresponding to the soft, the loud, and the shout-trained models, were applied to generate three new word models. The variances in these models were scaled according to Fig.1(c). In recognition, the four models (including the original normal model) were treated independently and equally; in effect, the computation for HMM recognition was quadrupled.

The recognition error rates of these experiments are listed in Table 11. The error rate reductions relative to the baseline system seem quite promising given the simplicity of the compensation technique.

The next section discusses a variation of the above technique--the hypothesis-driven stress compensation.

CEPSTRAL DOMAIN STRESS COMPENSATION - A HYPOTHESIS DRIVEN APPROACH

It is the high cost of increased computation and the uncertainty about training-style sufficiency and efficiency that prompted us to search for alternatives. As a result of this effort, the hypothesis-driven cepstral mean compensation technique, which adapts to the input speech and to the hypothesized reference word, was developed. Fixed multistyle variance compensation has been found beneficial for all styles and will be used in conjuction with the adaptive mean compensation.

In deriving this technique, we model the talker as an information source (Fig. 2) that puts out a sequence of deterministic cepstral vectors $\{c_t\}$.[2] Before the vectors are received by the decoder, we assume that they undergo two stages of contamination.

Stage 1

A sequence of independent identically distributed (i.i.d.) random vectors $\{\delta_t\}$ is added to the cepstral sequence $\{c_t\}$ to create a new sequence $\{u_t\}$

$$u_t = c_t + \delta_t \qquad (1)$$

The sequence $\{\delta_t\}$ models the randomness of speech cepstral parameter outputs; its elements are assumed to be normally distributed with zero mean vector and diagonal covariance matrix.

[1] In a different data base we have observed inflated variance scaling only in the low order coefficients.

[2] The subscript t is an index of time.

Stage 2

A deterministic but unknown vector χ is added to the sequence $\{u_t\}$ to create the observation sequence $\{v_t\}$

$$v_t = u_t + \chi \tag{2}$$

The vector χ is the additive "stress" component. It is assumed to have the functional form [see Fig. 1(a)]:

$$\chi_i = a\, e^{-b(i-1)} \tag{3}$$

and is further assumed to remain unchanged within a word interval.

Given a sequence of observations v_t, t=1,2,...,T we have developed a procedure for estimation, based on maximum likelihood principles, of the parameters a and b in Equation (3).

The procedure consists of two steps:

Step 1 (Estimating χ_i)

The probability density function of v_i, the i^{th} component of the observation vector, is given by

$$f(v_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp - \frac{(v_i - c_i - \chi_i)^2}{2\sigma_i^2} \tag{4}$$

where c_i and χ_i are the ith components of the cepstral vector and the "stress" vector, respectively.

Given a set of independent observations $\{v_t\}$, t=1,...,T, the maximum likelihood estimate of χ_i is given by

$$\bar{\chi}_i = \frac{1}{T} \sum_{t=1}^{T} (v_{it} - c_{it}) = \frac{1}{T} \sum_{t=1}^{T} v_{it} - \frac{1}{T} \sum_{t=1}^{T} c_{it} \tag{5}$$

We replace the sample average of c_i, which is not observable, by the expected average value, derived from the word hypothesis:

$$\bar{c}_i = E\left[\frac{\sum_n \tau_n c_{in}}{\sum_m \tau_m}\right] = \sum_n E\left[\frac{\tau_n}{\sum_m \tau_m}\right] c_{in} \tag{6}$$

where the τ_n's are a set of mutually independent discrete random variables whose values represent the dwell time in each of the n nodes, and the summations are over all speech nodes.

Since a closed formula for $E\frac{\tau_n}{\sum \tau_m}$ has not been found, we use an approximation using up to the second-order moments. Let

$$\gamma_n = \sum_{\substack{m=1 \\ m \neq n}}^{N} \tau_m \tag{7}$$

$$g(\tau_n, \gamma_n) = \frac{\tau_n}{\tau_n + \gamma_n} \tag{8}$$

then τ_n and γ_n are independent and the expectation can be approximated by

$$E[g(\tau_n,\gamma_n)] \approx g(\bar{\tau}_n,\bar{\gamma}_n) + \frac{1}{2}\sigma_{\tau_n}^2 \frac{\partial^2 g}{\partial \tau_n^2} + \sigma_{\gamma_n}^2 \frac{\partial^2 g}{\partial \gamma_n^2} \tag{9}$$

$$= \frac{\bar{\tau}_n}{\bar{\tau}_n + \bar{\gamma}_n} + \frac{\bar{\tau}_n \sigma_{\gamma_n}^2 - \bar{\gamma}_n \sigma_{\tau_n}^2}{(\bar{\tau}_n + \bar{\gamma}_n)^3}$$

with the means and variances given by

$$\begin{cases} \bar{\tau}_n = \frac{1}{P_n} \\ \bar{\gamma}_n = \sum_{\substack{m=1 \\ m \neq n}}^{N} \frac{1}{P_m} \\ \sigma_{\tau_n}^2 = \frac{(1-P_n)}{P_n^2} \\ \sigma_{\gamma_n}^2 = \sum_{\substack{m=1 \\ m \neq n}}^{N} \frac{(1-P_n)}{P_m^2} \end{cases} \tag{10}$$

The estimation formula (5) becomes

$$\bar{\chi}_n = \frac{1}{T} \sum_{t=1}^{T} v_{nt} - \sum_{n=1}^{N} E[g(\tau_n,\gamma_n)]\, c_n \tag{11}$$

In Equation (11) the first sum is over the observed cepstral coefficient sequence, and the second sum is over the nodes of the hypothesized model. Therefore, we refer to this technique as a hypothesis-driven technique.

Step 2 (Smoothing χ_i)

After $\chi_1,\ldots\chi_{12}$ are estimated, we fit Equation (3) to them. A least-mean-square fit requires numerically solving a set of nonlinear equations. A less computationally intensive and yet more robust fit (i.e., one which is less susceptible to the effect of outlying data), is given by fitting exponential functions to all pairs $\{\chi_i,\chi_j\}$, $i \neq j$, or a subset of these pairs, and then by averaging magnitudes and time constants of the fits. We have chosen to fit the pairs that contain χ_1 and one of χ_2,χ_3,χ_4 and χ_5, namely, $\{\chi_1,\chi_j\}$, j=2,3,4,5. Therefore,

$$b_j = \begin{cases} -\ln \dfrac{X_j}{X_1} & , \; X_1 X_j > 0 \text{ and } |X_1| > |X_j| \\ 0 & , \; \text{otherwise} \end{cases}$$

$$a_j = \begin{cases} X_1 & , \; b_j \neq 0 \\ 0 & , \; \text{otherwise,} \end{cases} \qquad (12)$$

a and b are the average of non-zero a_j's and b_j's.

Given the cepstral vectors of a test token and the Hidden Markov word model for a reference, the procedure for the adaptive cepstral compensation and recognition is described as follows:

Step 1: Compute a set of stress components [c.f. Eq. (11)].

Step 2: Smooth the stress components by fitting an exponential function to them [c.f. Eqs. (3) and (12)].

Step 3: Subtract the values of the exponential function from the cepstral vectors of the test token.

Step 4: In recognition, perform likelihood tests using the compensated test tokens.

In Table III we summarize the recognition error rates when the hypothesis-driven stress compensation is applied to the "simulated stress" data base. For comparison, the error rates of the baseline and of multimodel compensation are also included. This technique has also been applied to a more advanced 14-node, fixed-variance HMM system [1] whose parameters contain cepstral coefficients as well as differential cepstral coefficients. Because cepstral variances are fixed in this recognizer, no variance scaling is performed. The recognition results, with and without cepstral compensations, are listed in Table IV.

A confidence interval analysis indicates that the improved error rates in Tables III and IV, 6.2% and 1.9%, lie well outside the 95% confidence intervals of the unimproved error rates, 13.9% and 2.5%. Therefore, our experimental results are statistically significant.

CONCLUSION

Spectral tilt has been found to vary significantly for speech spoken in stressful talking environments. We studied the statistical variations of cepstral coefficients embedded in the framework of Hidden Markov models and that the observed changes in cepstral mean values, from normal exponential type of spectral tilt. A simple and efficient compensation technique, the hypothesis-recognition experiments yielded significant reduction in error rate.

ACKNOWLEDGEMENT

I would like to acknowledge that Dr. D. B. Paul provided the baseline HMM recognizer and conducted the experiments presented in Table I, and that multistyle training techniques discussed here were developed and tested by Dr. R. P. Lippmann and Dr. D. B. Paul.

REFERENCES

[1] D.B. Paul, "A Speaker-Stress Resistant HMM Isolated-Word Recognizer," ICASSP '87 (April 1987).

[2] Y. Chen, "Cepstral-Domain Talker Stress Compensation for Robust Speech Recognition" M.I.T. Lincoln Laboratory Technical Report 753 (November 1986)

[3] P.K. Rajasekaran, G.R. Doddington and J.W. Picone, "Recognition of Speech Under Stress and in Noise," Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing (1986).

[4] R.P. Lippmann, M.A. Mack, and D.B. Paul, "Multi-Style Training for Robust Speech Recognition Under Stress," J. Acoust. Soc. Am., Supplement 1, 79, S95 (1986).

[5] R.P. Lippmann, E.A. Martin, and D.B. Paul, "Multistyle Training for Robust Isolated-Word Speech Recognition," ICASSP '87 (April 1987).

*This work was sponsored by the Defense Advanced Research Projects Agency.

The views expressed are those of the author and do not reflect the official policy or position of the U.S. Government.

TABLE I
SUBSTITUTION RATE (PERCENT): A COMPARISON OF NORMAL- AND MULTISTYLE-TRAINED HMM RECOGNIZERS

Condition	Norm	Fast	Loud	Noise	Soft	Shout	*** Avg5	Avg6
Baseline HMM*	1.0	6.1	29.1	19.6	13.5	86.4	13.9	25.9
Multistyle**	0.5	5.6	5.1	2.1	5.8	43.6	3.8	10.5

* The baseline system was trained with 5 normally spoken word tokens per talker and tested on 10,080 test tokens.

**The multistyle-trained system was trained on 11 style speech tokens per talker and tested on 5,040 test tokens.

***The Avg5 is an average of the error rates of all styles except shout.

TABLE II
SUBSTITUTION RATE (PERCENT):
A COMPARISON OF FIXED STRESS COMPENSATION

Condition	Norm	Fast	Loud	Noise	Soft	Shout	Avg5	Avg6
Single Model	1.2	4.6	15.2	12.2	15.4	79.5	9.7	21.4
Multimodel	1.0	4.2	12.1	6.7	5.5	68.7	5.9	16.4

TABLE III
SUBSTITUTION RATE (PERCENT):
A COMPARISON OF MULTIMODEL FIXED STRESS COMPENSATION WITH HYPOTHESIS-DRIVEN STRESS COMPENSATION

Condition	Norm	Fast	Loud	Noise	Soft	Shout	Avg5	Avg6
Baseline HMM	1.0	6.1	29.1	19.6	13.5	86.4	13.9	25.9
Multimodel	1.0	4.2	12.1	6.7	5.5	68.7	5.9	16.4
Hypothesis-Driven	0.9	4.7	12.7	7.0	5.7	72.4	6.2	17.2

TABLE IV
SUBSTITUTION RATE (PERCENT):
AN ADVANCED HMM RECOGNIZER

Condition	Norm	Fast	Loud	Noise	Soft	Shout	Avg5	Avg6
Without Compensation	0.4	1.7	3.4	2.9	4.4	49.8	2.5	10.4
With Compensation	0.4	1.7	3.4	1.4	2.4	45.3	1.9	9.0

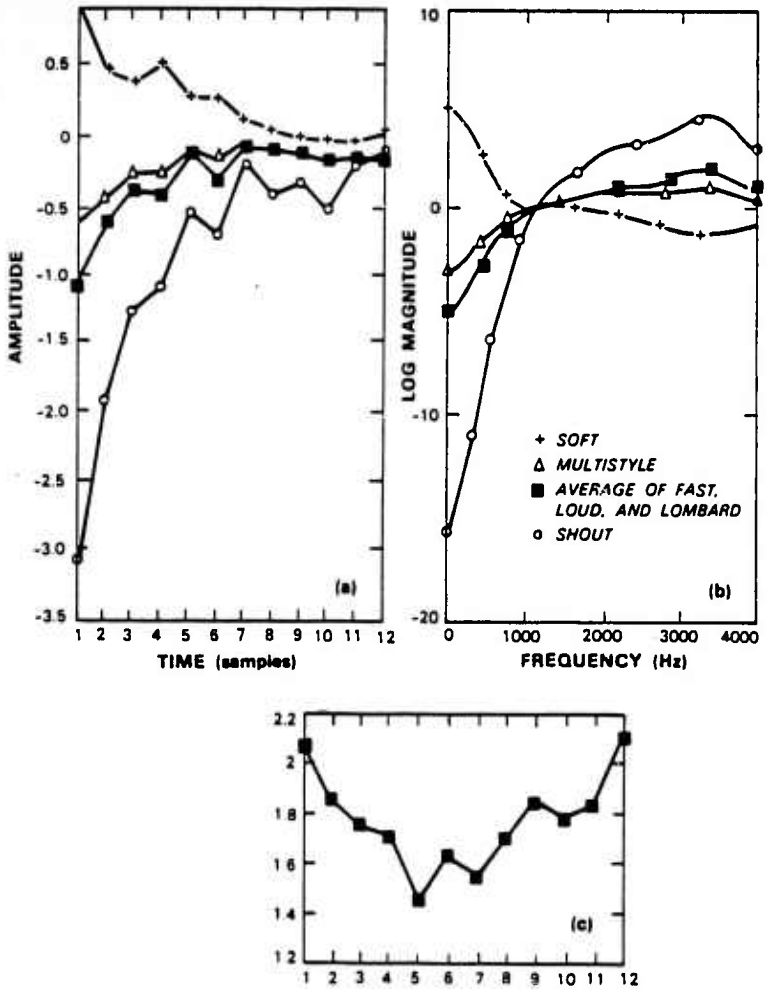


Fig. 1. Variations of cepstral coefficient compared to normally spoken words. (a) Difference of mean (style minus normal), (b) spectra of differences of mean, and (c) ratio of variance (multistyle normal).

This paper appears in the Proceedings of ICASSP 87.

MULTI-STYLE TRAINING FOR ROBUST ISOLATED-WORD SPEECH RECOGNITION

Richard P. Lippmann, Edward A. Martin, Douglas B. Paul

Lincoln Laboratory, Massachusetts Institute of Technology
Lexington, Massachusetts 02173-0073

ABSTRACT

A new training procedure called multi-style training has been developed to improve performance when a recognizer is used under stress or in high noise but cannot be trained in these conditions. Instead of speaking normally during training, talkers use different, easily produced, talking styles. This technique was tested using a speech data base that included stress speech produced during a workload task and when intense noise was presented through earphones. A continuous-distribution talker-dependent Hidden Markov Model (HMM) recognizer was trained both normally (5 normally spoken tokens) and with multi-style training (one token each from normal, fast, clear, loud, and question-pitch talking styles). The average error rate under stress and normal conditions fell by more than a factor of two with multi-style training and the average error rate under conditions sampled during training fell by a factor of four.

INTRODUCTION

The performance of current recognition systems often degrades dramatically as a talker's speech characteristics change with time, when a talker is under normal levels of workload or psychological stress, and when a talker is in a high noise environment. New techniques to prevent this degradation have been developed and tested with a number of data bases, including a new Lincoln stress-speech data base. In this paper we first review results obtained with this speech data base and then provide detailed information on the effects of multi-style training. Other papers in this proceedings describe discriminent analysis [1] and cepstral stress compensation [2] and present results obtained with another speech data base [3].

Lincoln Stress-Speech Data Base

The Lincoln stress-speech data base includes words spoken with eight talking styles (normal, slow, fast, soft, loud, clear enunciation, angry, question pitch) and under three stress conditions. A difficult motor-workload task [4] was used to create easy (cond50) and more difficult (cond70) workload stress conditions that emulate the type of workload stress experienced when driving a car or flying an airplane. A third stress condition was created by presenting 85 dB SPL of speech-shaped noise through earphones. This produces the so-called Lombard effect [8] where a talker speaks louder and often more clearly when in noise. This is the main cause for recognizer degradation in noise in situations where an acoustically-shielded close-talking microphone minimizes the effect of additive noise. The data base vocabulary contained 35 difficult aircraft words with acoustically similar subsets such as go, hello, oh, no, and zero. A total of 11,340 tokens were obtained from 9 male talkers during three sessions per talker spanning a four week period.

HMM Recognizer

The baseline continuous-distribution HMM recognizer described in [4] was used for all experiments. It is a left-to-right isolated-word recognizer with multivariate Gaussian distributions and diagonal covariance matrices where observations consist of centisecond mel-scale cepstral parameters. Unless otherwise stated, all results were obtained using 10-node word models created using five training tokens per word with the forward-backward algorithm [5] and using the Viterbi algorithm [5] during recognition.

RESULTS WITH LINCOLN STRESS-SPEECH DATA BASE

Figure 1 presents an overview of results in rough chronological order obtained using a number of different techniques with the Lincoln stress-speech data base. The initial error rate, averaged over all conditions excluding the most difficult angry condition, was 17.5%. A similar high error rate was obtained with a new, high performance, commercial recognizer. Poor performance for the initial Lincoln system and the commercial system was caused by the difficult vocabulary and stress conditions and by the fact that only normally-spoken speech was used in training. The initial Lincoln recognizer was the baseline system with variance limiting [4] which limits the variance estimates obtained during forward-backward training to be above a specified lower limit. The high initial error rate was more than halved to 6.9% using multi-style training. In this case, the five tokens used during training were taken from the normal, fast, clear, loud, and question-pitch talking styles instead of only from the normal style. Multi-style training halved the

error rate with no increase in computation requirements.

The next large reduction in error rate (from 6.9% to 3.2%) was obtained by doubling the number of parameters used in the observation vector. The original vector of 16 cepstral parameters was supplemented with 16 additional differential parameters which were the differences between the current 16 parameters and the parameters computed 20 ms earlier. This differential parameter technique was also recently used by [6]. It reduces the error rate, but also doubles the recognition computation requirements. The next large decrease in error rate (from 3.2% to 1.6%) was obtained by using grand-variance estimates. Instead of estimating the variance of each of the 32 observation parameters separately for each node of every word model, the grand variance of each observation parameter was estimated once across all word models and all nodes during training. Using grand variances reduces the degradation in performance caused by using a statistical model that is too complex for the amount of training data. This result reinforces past results that demonstrate the necessity of matching the complexity of a model to the amount of training data [7]. Using grand variances halved the error rate while simultaneously decreasing recognition computation requirements. The final large reduction in error rate (1.6% to 1.0%) was obtained using the two-stage discriminant analysis system described in [1]. This system focusses attention on those parts of often confused words that are most different and reduces the error rate with only a slight increase in recognition computation requirements. The final system with a 1% error rate across many stress/style conditions is a usable, practical, robust recognizer that could be used for a variety of speech-recognition tasks.

EFFECTS OF MULTI-STYLE TRAINING

More details on the effects of multi-style training from the experiments described above are presented in Figs. 2 to 4. Figure 2 compares results with normal and multi-style training for the six novel conditions not sampled during training as well as for normally-spoken speech. These are representative results for the situation where a recognizer cannot be trained under live stress conditions. The percentage error rate averaged over all nine talkers is presented for normal speech, for speech spoken slowly, for the easy (cond50) and the more difficult (cond70) workload task, for soft speech, for speech produced in noise (Lombard) and for angry speech. Multi-style training reduces the error rate substantially for all conditions. The average error rate over all conditions fell by more than a factor of two from 20.7% to 9.8%. The drop in error rate is large (% to 2.9%) even for normally spoken words and greatest for the Lombard and angry conditions.

Figure 3 shows the results when the recognizer was tested under the same conditions sampled during training. Here, the average error rate over all conditions fell by a factor of four from 18.4% to 4.6%. It should be noted that in these and other experiments, training word tokens were never used during testing.

Further experiments were performed to determine whether more effective subsets of five styles could be found and whether fewer than five different styles could provide large improvements. These experiments suggest that the five styles selected are more effective than other subsets of the eight styles in the stress-speech data base and that all five different styles are required for best performance with multi-style training. Further experiments have also been performed to explore the effects of multi-style training with more advanced HMM isolated-word talker-dependent recognizers. We have found that multi-style training always improves overall performance. For example, the error rate for an advanced recognizer with differential parameters, grand-variance estimates, 14 nodes, and five training tokens, drops from 3.2% to 1.4% with multi-style training.

One surprising result evident in Figs. 2 and 3 is that the error rate drops for normal speech when the recognizer is trained on non-normal training tokens. This is caused by day-to-day variability in normal speech as demonstrated in Fig. 4. Figure 4 presents the error rate with normal and multi-style training for normal speech recorded in the first, second, and third recording sessions. As can be seen, multi-style training and normal training produce similar results in session one, but multi-style training is superior in sessions two and three. These results demonstrate that multi-style training can compensate for variability in normal speech over time, and that five normal training tokens recorded in one session are less representative of normal tokens recorded one to three weeks later than five multi-style tokens.

DISCUSSION

Multi-style training improves performance for the novel stress conditions because: (1) the forward-backward training algorithm and statistical decoding focuses attention on spectral/temporal regions that are consistent across styles and (2) speech samples are presented during training that are similar to those that occur during testing. For example, loud speech is similar in many ways to speech produced under the Lombard condition. The improvement in performance with conditions sampled during training was greater than the improvement with novel untrained condition for this second reason.

A careful analysis of differences between word models obtained using normal and multi-style training and of recognizer confusions indicated that improvements are caused by two main mechanisms. First, estimates of the mean and variance of the cepstral parameters used in HMM word models are more representative of those observed during testing with multi-style training. This is illustrated in Fig. 5. The left side of this figure presents the difference

between multi-style and normally trained cepstral mean estimates and the right side presents the ratio of multi-style over normally trained cepstral variance estimates. Data are averaged over all talkers and all word models. As can be seen, the lower-order cepstral mean estimates are reduced with multi-style training. This compensates for spectral tilt (presumably caused by narrower glottal pulses) which is characteristic of much of the stress speech in the Lincoln data base. Variance estimates for lower-order cepstral coefficients are also higher with multi-style training. This weights these cepstral coefficients less heavily during recognition because they are more variable across stress and style conditions.

A second mechanism that leads to better performance with multi-style training is that word models are richer and provide a better description of perceptually important acoustic events that are present across talking styles. This mechanism was discovered by examining spectrograms created from normal and multi-style HMM word models for those models that caused major confusions. Spectrograms were created by plotting the average spectrum at each node with duration equal to the average node residency time. For example, Fig. 6. contains spectrograms generated from HMM word models for the word "break". The left spectrogram was generated from a normally-trained word model and the right one was generated from a multi-style model. Numbers indicate the HMM node number used to generate each spectra. In these and all experiments, the end nodes (nodes numbered 0 and 9 in Fig. 6) are anchors that match background noise. The large ticks in Fig. 6 are at 100 ms intervals, the lower curve plots overall energy, and the frequency scale extends to roughly 6 kHz. As can be seen, the multi-style word model contains the optional release for the final /k/ and provides a clearer description of formant transitions. Examination of many other word-model spectrograms showed that multi-style word models generally contain more of the important acoustic-phonetic cues used in spectrogram reading than normally-trained models.

SUMMARY

A new training procedure called multi-style training was developed and tested with a stress-speech data base. It improves performance substantially under stress and with different talking styles, and can be used when a recognizer cannot be trained under live stress conditions. It also improves performance under normal conditions by compensating for normal day-to-day speech variability.

REFERENCES

[1] E. A. Martin, R. P. Lippmann, and D. B. Paul, "Two-Stage Discriminant Analysis for Improved Isolated-Word Recognition," ICASSP'87, Dallas, TX, April 1987.

[2] Y. Chen, "Cepstral Domain Stress Compensation for Robust Speech Recognition," ICASSP'87, Dallas, TX, April 1987.

[3] D. B. Paul, "A Speaker-Stress Resistant HMM Isolated-Word Recognizer," ICASSP'87, Dallas, TX, April 1987.

[4] D. B. Paul, R. P. Lippmann, Y. Chen, and C. J. Weinstein, "Robust HMM-Based Techniques for Recognition of Speech Produced Under Stress and in Noise," Proc. Speech Tech 86, pp. 241-249, New York, NY, April 1986.

[5] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition," BSTJ, 62, pp. 1035-1074, April 1983.

[6] E. L. Bocchieri, and G. R. Doddington, "Frame-Specific Statistical Features for Speaker Independent Speech Recognition," IEEE Trans. Acoust. Speech, and Signal Processing, ASSP-34, pp. 755-764, August 1986.

[7] G. F. Hughes, "On the Mean Accuracy of Statistical Pattern Recognizers," IEEE Trans. Info. Theory, IT-14, pp. 53-63, January 1968.

[8] E. Lombard, "Le Signe de l'Elevation de la Voix," Ann. Maladies Oreille, Larynx, Nez, Pharynx, 37, 1911.

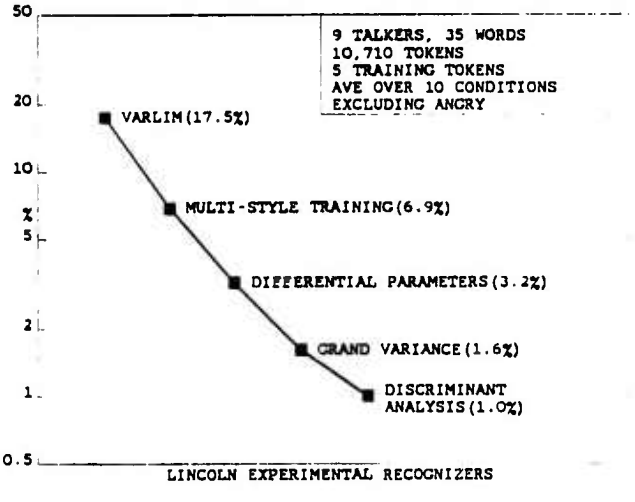


Fig. 1. Substitution errors with Lincoln stress-speech data base.

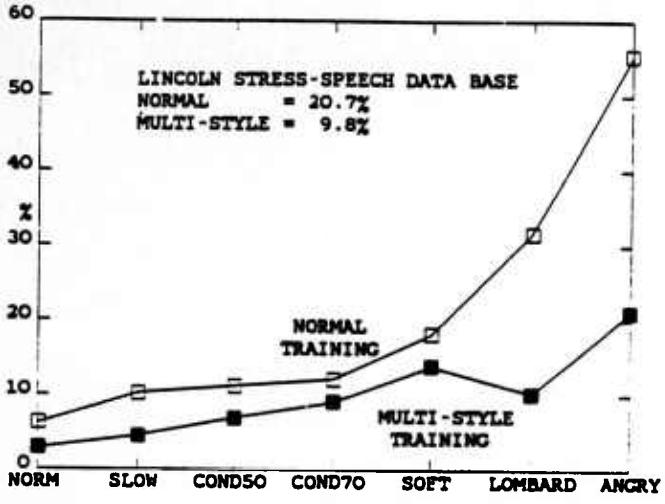


Fig. 2. Percent errors for novel untrained conditions.

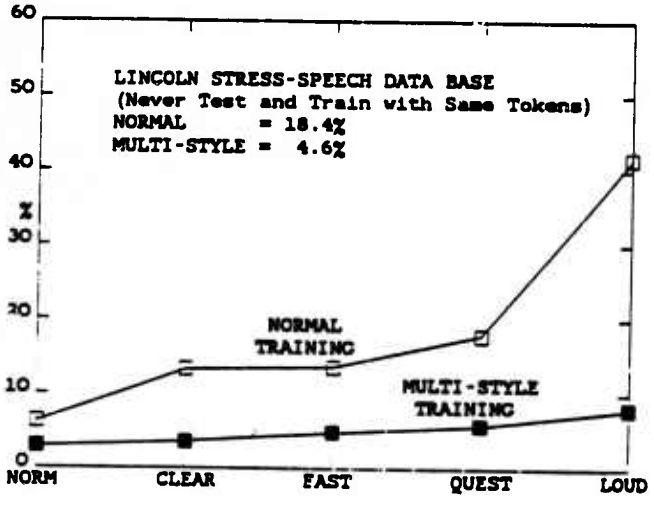


Fig. 3. Percent errors for conditions used during training.

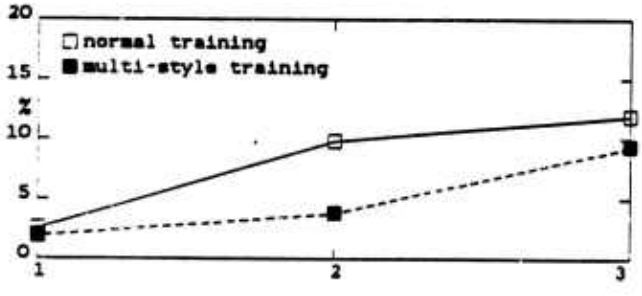


Fig. 4. Percent errors for normal speech across recording sessions.

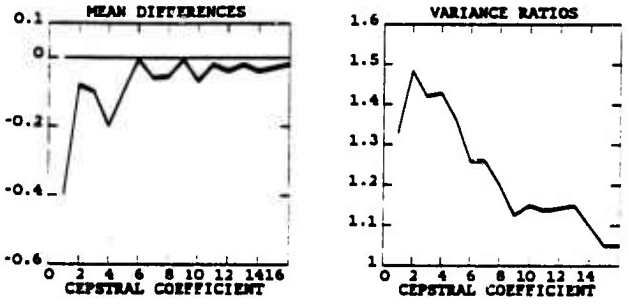


Fig. 5. Differences between means (A) and ratios of variances (B) from HMM models created using multi-style training and normal training.

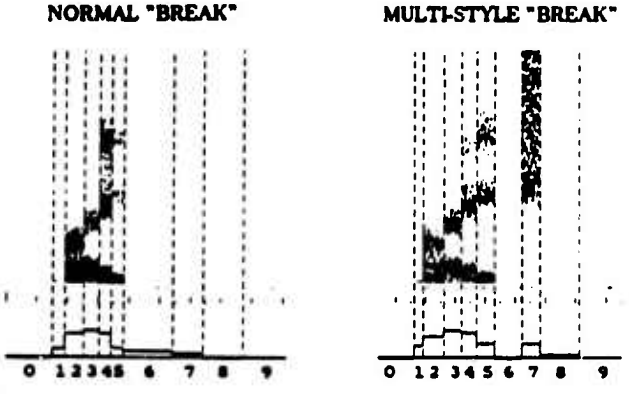


Fig. 6. Spectrograms of the word "break" created from normally-trained HMM word model (A) and multi-style-trained model (B).

This work was sponsored by the Defense Advanced Research Projects Agency.

The views expressed are those of the authors and do not reflect the official policy or position of the U.S. Government.

This paper appears in the Proceedings of ICASSP 87.

TWO-STAGE DISCRIMINANT ANALYSIS FOR IMPROVED ISOLATED-WORD RECOGNITION

Edward A. Martin, Richard P. Lippmann, Douglas B. Paul

Lincoln Laboratory, Massachusetts Institute of Technology
Lexington, Massachusetts 02173

ABSTRACT

This paper describes a two-stage isolated word speech recognition system that uses a Hidden Markov Model (HMM) recognizer in the first stage and a discriminant analysis system in the second stage. During recognition, when the first-stage recognizer is unable to clearly differentiate between acoustically similar words such as "go" and "no" the second-stage discriminator is used. The second-stage system focuses on those parts of the unknown token which are most effective at discriminating the confused words. The system was tested on a 35 word, 10,710 token stress speech isolated word data base created at Lincoln Laboratory. Adding the second-stage discriminating system produced the best results to date on this data base, reducing the overall error rate by more than a factor of two.

1. INTRODUCTION

A two-stage discriminant analysis system has been developed to address some of the problems generally encountered in current Hidden Markov Model (HMM) isolated word recognition systems. These problems include: (1) the effects of limited training data are not explicitly taken into account; (2) the correlation between adjacent observation frames is incorrectly modeled; (3) durations of acoustic events are poorly modeled; and (4) features which might be important in discriminating only among specific word pairs, or sets of words, are not easily incorporated into the system without degrading overall performance. The two-stage system uses new statistical techniques that explicitly account for the limited amounts of training data available for talker-dependent recognition. The second-stage system focuses its attention on those parameters in the models which are most effective in discriminating between words which achieve similar scores in the first-stage HMM system. A similar two-stage system was developed by Rabiner and Wilpon [9]. That system was developed in the context of a Dynamic Time Warping (DTW) rather than an HMM system, and also did not explicitly take into account the effects of limited training data. Another approach to the focus-of-attention problem in discrimination is presented in [6]. The new two-stage discriminant system described here was developed as part of a larger effort aimed at reducing the effects of stress on robust speech recognition systems [1,4,7,8].

2. OVERVIEW

The structure of the two-stage system is shown in Fig. 1. Each word model in the first-stage HMM recognizer is created using forward-backward training and training tokens for that word [3]. A detailed description of the HMM recognizer which is used is given in [4,7,8]. The recognizer uses a continuous-distribution speaker-dependent 10-node HMM model with 16 cepstral coefficients that are assumed to be jointly Gaussian and independent. The HMM model is trained using five tokens per word with multi-style training [4,7] and variance limiting [7,8].

The second-stage discriminant system calculates statistics, on the cepstral parameters and on selected additional parameters (see below), for each word model, vocabulary word, and node by decoding training tokens of all words using the Viterbi algorithm with HMM word models for all words. This "cross-word training" provides additional statistical information which is not available in standard HMM training, where each word model is trained only on samples of that word. During recognition, discriminant decisions are based on likelihood-ratio comparisons among all word pairs in the top N words from the HMM system. The comparisons assume that discriminant statistics are jointly Gaussian and independent. In addition, a new technique called "sifting" is applied, which uses a statistical "T" test to focus attention only on discriminant statistics that were judged from the training data to be statistically different, for specific word pairs.

The discriminant system was tested using the Lincoln Laboratory stress-speech data base [4,8]. This includes 10,710 words from nine talkers producing 35 acoustically-similar aircraft words spoken normally, under workload stress, in noise presented over earphones, and with seven different talking styles.

3. DISCRIMINANT TRAINING

Figure 1 illustrates the flow of data for the discriminant training process. During training, all tokens of all training words were decoded by all of the first stage HMM word models. Each decode resulted in a segmentation. From this procedure a statistical description was obtained

characterizing the distribution of observations that were assigned to each node of each word model, given a specific input word. Each estimated distribution was modeled as Gaussian. This information was then stored as two four-dimensional arrays: a mean and variance array indexed by word model, input word, node within the model, and parameter. Given a token segmented by a word model, these statistics were then available to be used, during recognition, to calculate a likelihood-ratio between any two hypothesized input words.

4. DISCRIMINANT RECOGNITION

During recognition, unknown tokens were first passed through the HMM system. Likelihood scores from the Viterbi algorithm were calculated for each word model. In cases where scores for two models, say for words A and B, were clearly better than all other models yet were very similar to each other, the second stage system was used. During the HMM pass, segmentation by each word model assigned each input observation to a specific node in that model. Per-node observations were used with the discriminant training statistics to seperately calculate the likelihood-ratio between the inputs being A and B given the segmentation from both the A and B word models.

An effort was made to seperate the scoring based on duration information from other aspects of the scoring. To implement this, likelihood-ratio scores for any input token were calculated on a per-node basis rather than on a per-observation basis. This was achieved by first calculating the likelihood-ratio based on all observations in a node, then normalizing this score by the number of observations assigned to that node. The advantage of this scheme was two-fold: first, it reduced the weighting of certain nodes which might dominate in the final score because of the large number of observations assigned to those nodes, and secondly, it eliminated the assumption made with per-observation scoring that all observations are statistically independent. On the contrary, this "per-node" scheme assumed a very strong correlation between observations assigned to the same node.

It should be observed that this per-node scoring technique removes duration information from the scoring. This was desirable since it enabled the duration information to then be explicitly modeled and included as a separate feature into the scoring mechanism. To facilitate this, two more arrays were generated during the discriminant training procedure described above. A mean and variance array were generated modeling the number of observations assigned to each node by a word model given each input word. This information was stored as two three-dimensional arrays indexed by word model, node in the model, and input word.

5. EXPERIMENTS

Since two likelihood-ratio scores were calculated for each pair-wise discrimination, corresponding to the segmentation arising from the pair of word models, a scheme had to be devised to account for possible disagreement from these two scores. For initial experiments it was decided that if discriminant scores disagreed, the decision would simply be deferred back to the original scores from the HMM system.

A decision also had to be made on criteria for deciding when the second-stage should be used, and how many of the top candidate words should be considered. To simplify initial experiments a hard threshold was established on the difference between the top two HMM word scores. If the difference between the two leading scores exceeded the threshold, the second stage was not used. It was later found that results were relatively insensitive to changes in this threshold. Discriminations were limited initially to consider only the top two candidate words.

The HMM system selected as the first stage was at the time, the best system tested on the Lincoln database, achieving an error rate of 7.7%. A more detailed description of this system is included in [4] and [7].

5.1 Estimated Variance

The first experiment with the two-stage system used all the cepstral parameters, as well as the duration and energy parameters, in the second-stage discriminator. Performance of this system was mediocre. The overall error rate fell from 7.7% with the basic HMM system to 7.4% with the two-stage system. It was suspected that part of the reason for the disappointing performance might be that only a subset of the parameters contributed positively to discrimination. To investigate the effectiveness of individual parameters as discriminators another experiment was performed which used only a single parameter in all second stage discriminations. This experiment was repeated for each available parameter. These included the sixteen cepstral coefficients from the HMM system, a relative energy measurement and a node duration measurement. Results using only the two best parameters (duration and relative energy), each individually, showed improvements over the previous experiment, where all parameters were included in the discrimination. An partial explanation of this result, it should be noted that when a very complex model is made for a system and very limited training data is available to characterize it, statistical noise from poor estimation can degrade performance. In the above experiment, the model was simplified to better match the amount of training data available. Less statistical noise was then introduced to the scoring, and because of this, overall system performance improved. The overall error rate dropped from 7.4% to 7.0% using only a single parameter (the duration parameter) in the second stage.

5.2 Grand Variance

The next experiment attempted to extend this concept. Instead of using the large variance array generated during training for the observation parameters, grand variance estimates were used [4,7]. The grand variance is a variance estimate for a single cepstral coefficient parameter sampled over all word models, nodes, and input words. By including grand variance estimates, the second-stage model was simplified and the number of samples used to characterize each variance estimate was greatly increased. The error rate dropped with this scheme from 7.0% to 6.3% with the single best parameter (the duration parameter). Using grand variance estimates, the second-stage system was modified to once again include all cepstral parameters in the discrimination. This change resulted in the error rate dropping from 6.3% to 4.6%. This result suggests that the poor performance found with the original second-stage system was due to poor variance estimates based on a very small number of samples.

5.3 Sifting

When an unknown token is compared to two word models of acoustically similar words, those parts of the models which correspond to identical acoustic events should make no contribution to a discrimination between the models. However, when limited training data is available to characterize these models, there will be slight differences in the estimated models for identical acoustic events. These differences can have an accumulated effect large enough to overwhelm the more important differences in the models which correspond to different acoustic events. The statistical T-test provides a technique for estimating the probability that two estimated distributions have identical underlying means [2]. A technique based on the T-test, which will be called "sifting," was used in the second stage to eliminate parameters from discrimination when the training data did not indicate significant differences in the underlying distributions of those parameters [5]. The effect of this process is to focus the discrimination on those parameters and nodes which correspond to the acoustical differences in the two models. Application of this technique resulted in a slight decrease in error rate from 4.6% to 4.5%. But more significant than the reduction in error rate achieved was the decrease in computation provided by the sifting technique. Approximately 50% of all parameters were excluded from the discrimination in this experiment, thus decreasing computation by a factor of two. Figure 3 illustrates the effect of sifting by indicating which parameters for a specific word pair are included in discrimination.

5.4 Added Features

This technique of sifting parameters enables features to be added to the system, which might only address a few specific confusions. These features can be added without degrading recognition for inputs where the features are not useful, since for those cases the sifting process should eliminate them from the scoring. To illustrate this, a new set of parameters was included in the discriminant scoring. These parameters were chosen to make use of longer term spectral changes in the speech signal. The mean value of all cepstral parameters were calculated for each node. The differences in these mean values between adjacent nodes were then included in the set of features used for discrimination. This effectively doubled the number of parameters used in the discriminant system from 18 to 35. This set of parameters was used along with T-testing, in another recognition experiment. Again, T-testing eliminated approximately half of the parameters from the scoring; and the error rate was slightly reduced from 4.5% to 4.4%.

5.5 Top Three Candidates

A final experiment which was performed included discrimination on the top three candidates as opposed to the top two as in previous experiments. The decision scheme was modified to account for this change and to lessen the incidence of the second-stage deferring back to the original HMM scores. This experiment resulted in the best performance at the time with the two-stage system reducing the error rate to 3.5%. A detailed look at the improvements this system showed over the HMM system alone is presented in Fig. 4. The discriminant system provided improved performance for all of the various speaking styles and stress conditions.

5.6 Tests with Advanced HMM System

After the series of experiments described above, an advanced first-stage HMM system [7] became available which produced an average error rate of 1.6% on the Lincoln stress speech database. Application of the best second-stage discriminator (as in Section 5.3) to this more advanced HMM system reduced the average error rate from 1.6% to 1.0% (see [4]).

5.7 Discussion of Confusion Statistics

Many of the errors from the first-stage system were a result of confusions between similar sounding words such as "eighty" and "eight," "fix" and "six," and "white" and "wide". Presumably these errors were attributable to the problems discussed earlier in this paper. Although some of these confusions persisted after implementing the second-stage system, most were eliminated, and the remaining errors were mostly scattered across many word pairs.

6. SUMMARY

A new two-stage recognition system has been developed which substantially reduces overall error rates and assists a recognition system in discriminating among acoustically similar words without compromising performance for the remainder of a vocabulary. Key characteristics of the system are that it specifically addresses problems caused by limited training data, and poor duration

models. The system also applies a statistically-based "lifting" technique to focus its attention on parameters which most effectively bring out differences in the words being discriminated.

REFERENCES

[1] Y. Chen, "Cepstral Domain Stress Compensation for Robust Speech Recognition," ICASSP'87, Dallas, TX, April 1987.

[2] R. Duda, P. Hart, Pattern Classification and Scene Analysis (John Wiley and Sons, NY, 1973).

[3] S. E. Levinson, L. R. Rabiner, M. M. Sondhi, "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition," The Bell System Technical Journal, Vol. 62, (4), pp. 1035-1074, 1983.

[4] R. P. Lippmann, E. A. Martin, D. B. Paul, "Multi-Style Training for Robust Isolated-Word Speech Recognition," ICASSP'87, Dallas, TX, April 1987.

[5] E. A. Martin, "A Two Stage Isolated Word Recognition System Using Discriminant Analysis," S.M. Thesis, M.I.T., June 1986.

[6] R. K. Moore, M. J. Russell, M. J. Tomlinson, "The Discriminative Network: A Mechanism for Focusing Recognition in Whole-Word Pattern Matching," ICASSP'83, Boston, MA, April 1983.

[7] D. B. Paul, "A Speaker-Stress Resistant HMM Isolated Word Recognizer," ICASSP'87, Dallas, TX, April 1987.

[8] D. B. Paul, R. P. Lippmann, Y. Chen, C. J. Weinstein, "Robust HMM-Based Techniques for Recognition of Speech Produced Under Stress and in Noise," Speech Tech 86, NY, April 1986.

[9] L. R. Rabiner, J. G. Wilpon, "A Two-Pass Pattern-Recognition Approach to Isolated Word Recognition," The Bell System Technical Journal, Vol. 60, (5), pp. 739-766, May 1981.

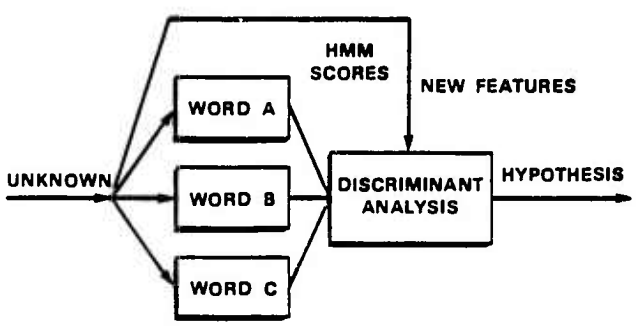


Fig. 1. Block diagram of two-stage system.

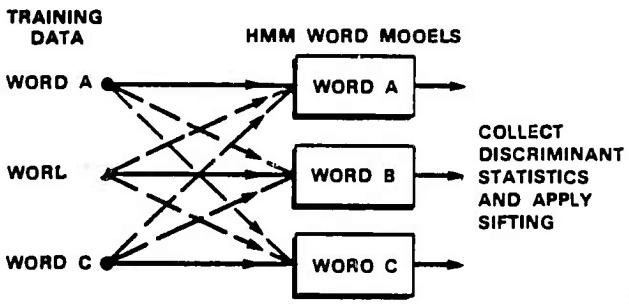


Fig. 2. Training for discriminator.

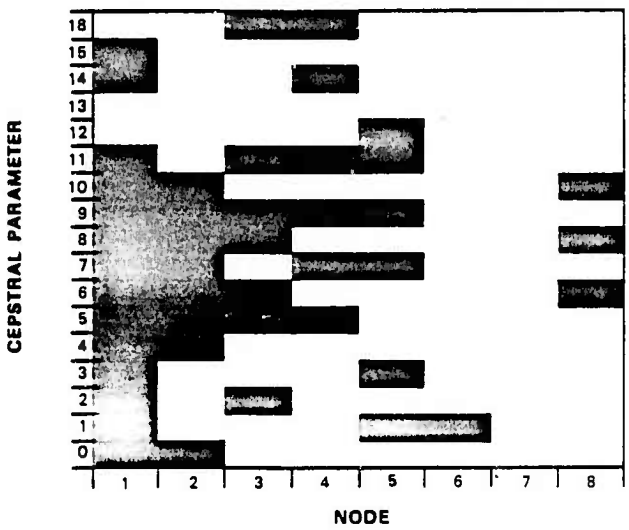


Fig. 3. Cepstral parameters used to discriminate the word models for "go" and "oh" are indicated with darkened regions. Most of the parameters used are concentrated toward the beginning nodes.

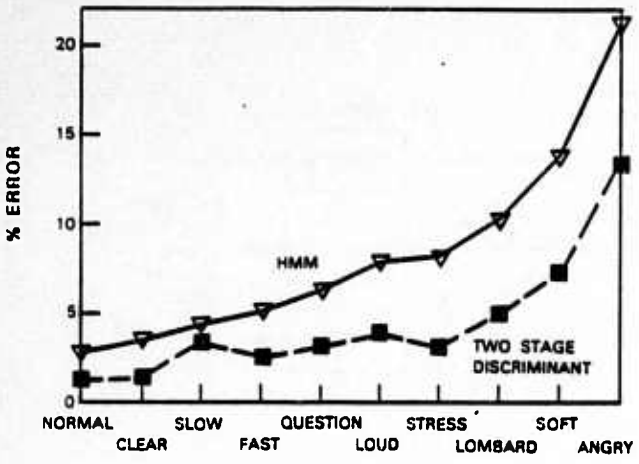


Fig. 4. Performance improvements on the Lincoln data base using an HMM recognizer alone and with discriminant analysis.

This work was sponsored by the Defense Advanced Research Projects Agency.

The views expressed are those of the authors and do not reflect the official policy or position of the U. S. Government.

This paper appears in the Proceedings of ICASSP 87.

THE DARPA TASK DOMAIN SPEECH RECOGNITION DATABASE

William M. Fisher

Texas Instruments Inc.
Computer Sciences Center
P.O. Box 225015, MS 238
Dallas, Texas 75266, USA
Tel. (214) 995-0394

ABSTRACT

This paper documents the DARPA Resource Management (RM) Task Domain speech data base, which is intended to be used in the evaluation of speech recognition systems that may incorporate a higher-level language model. The prompts, contributed by BB&N, were taken from a specific sub-language model. In addition to full sentence utterances, "spell mode" word spellings were recorded. For use primarily with speaker-independent recognizers, 57 utterances were recorded from each of 160 speakers; a speaker-dependent set of data is provided by recordings of 1012 utterances from each of 12 speakers. The speakers were selected, with the help of SRI, from among the 630 speakers recorded previously in the pan-dialectal TIMIT Acoustic-Phonetic data base. Recording formats and facilities were the same, with the exception of an improvement in suppression of background noise.

1. INTRODUCTION AND BACKGROUND

The RM task domain data base was designed during 1986, in collaboration with NBS, CMU, BB&N, and SRI. Both speaker independent and speaker dependent phases were segmented into training, development test, and evaluation test recordings. Digital tapes of the recordings were shipped to the National Bureau of standards (NBS), for further distribution to users, during the course of the data base collection.

The original plan was to give the speaker independent recordings priority, substantially completing them before starting on the speaker dependent recordings. Recording of the speaker independent part began on 10/16/86; its training phase was completed 11/20/86 and its development test phase was approaching completion in early December when the decision was made to deliver speaker dependent data and speaker independent data to users at about the same rates. Speaker dependent recording began on 12/10/86 and was given priority. Speaker independent development test data recording was finished on 12/17/86. training and development test sentence recordings for the first four of the twelve speaker dependent recordings were completed on 2/10/87; final completion date of all recording was 3/25/87 for speaker independent data and 3/26/87 for speaker dependent.

The purpose of the task domain data base is to provide speech data limited by a language model. The language model, developed at BB&N, covers utterances appropriate for a specific naval resource management task. TI received 2835 sentences generated from this model as a pool from which to draw prompts. A subset of 600 of these sentences had been hand picked at BB&N as training sentences; in the following explanations these may be referred to as the PJSENT1 sentences. The other 2235 RM sentences are the PJSENT2 sentences. Ten sentences from the same language model were selected at SRI as peculiarly appropriate for rapid phonetic adaptation. At TI these sentences were formatted to normal orthographic standards and used as prompts. 600 words from the vocabulary of the language model were selected and made into prompts for "spell-mode" readings. Subjects also re-read the two SRI "dialect" sentences that were used to calibrate dialect usage.

Subjects were recruited from the sample of 630 who had given speech earlier for the TIMIT Acoustic-Phonetic data base. Selection was guided by an analysis of the subjects' observed phonetic characteristics, made at SRI. 160 subjects were used in the speaker independent phase and 12 subjects in the speaker dependent phase.

2. STRUCTURE

The macrostructure of the data base is exhibited in the two figures below, Figure 1 for the speaker independent phase and figure 2 for speaker dependent. Subjects are arrayed vertically and utterances or sentence productions horizontally.

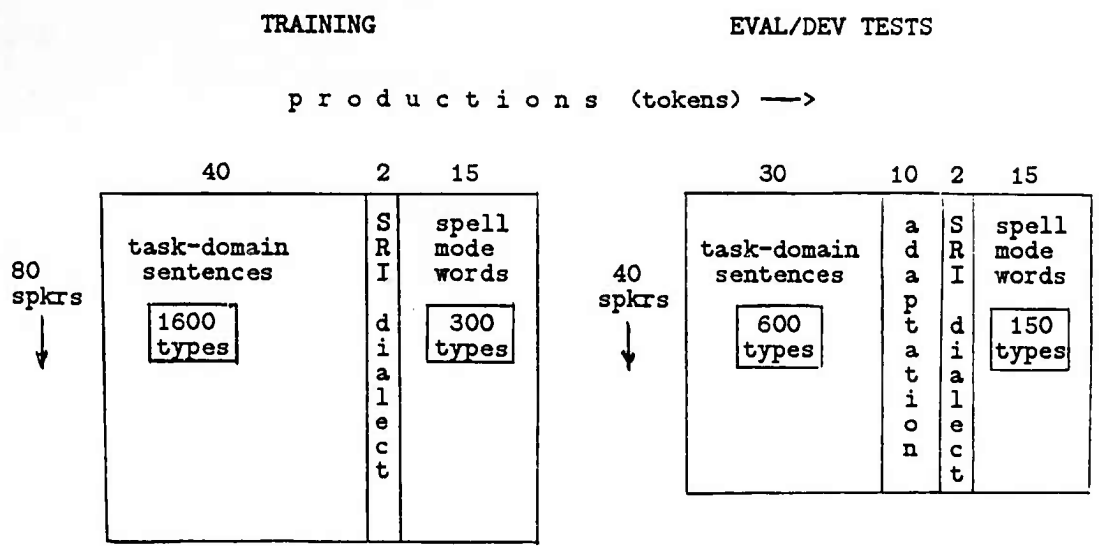


Figure 1. Speaker Independent Task Domain Data Base Layout

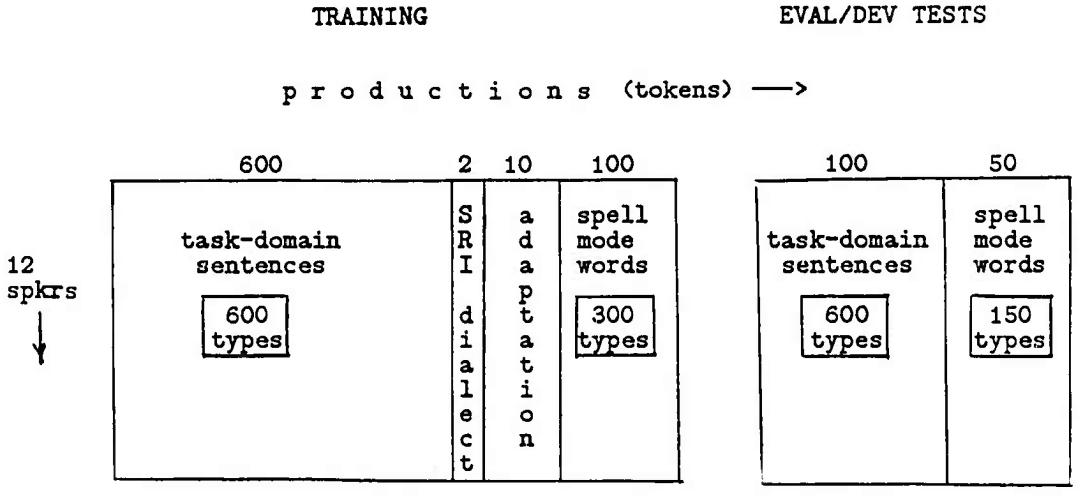


Figure 2. Speaker Dependent Task Domain Data Base Layout

Sentence I.D.	Significance
SR001 - SR600	600 training RM sentences from PJSENTS1
ST0001- ST2235	2235 other RM sentences from PJSENTS2
SA1 - SA2	2 SRI dialect sentences
SP001 - SP600	600 Spell-mode word sentences

Table 1. Sentence I.D. Key.

In the speaker independent training phase, each of 80 speakers read 40 RM sentences, 2 SRI dialect sentences, and 15 spell-mode words. In all, 1600 distinct RM sentences (sentence types as opposed to tokens) were available in this part, resulting in each sentence type having two tokens, being read by two subjects. The distribution of sentences to speakers was arbitrary, with the exception that no sentence was read twice by the same subject. Both SRI dialect sentences were read by each subject. Each speaker read 15 spell-mode words, yielding (80x15) 1200 productions, which were covered by a selection of 300 words. Each spell-mode word was thus read by 4 speakers.

The speaker independent development and evaluation test sets have identical form factors. In each, 40 speakers each read 30 RM sentences, the 10 rapid phonetic adaptation sentences, the 2 SRI dialect sentences, and 15 spell-mode words. 600 RM sentence types were randomly selected for each test and assigned to the 1200 available productions, as in the training phase. Similarly, 150 spell-mode words were selected and assigned to the 600 available spell-mode productions.

For speaker dependent training, each of the 12 subjects read each of the 600 PJSENT1 RM sentences, the 2 SRI dialect sentences, the 10 rapid phonetic adaptation sentences, and a selection of 100 spell-mode words. The 1200 spell-mode word readings thus produced were covered by a selection of 300 word types, resulting in four productions per word.

In the speaker dependent development and evaluation test sets, each of the same 12 speakers read a selection of 100 RM sentences and 50 spell-mode words. Two random selections of 600 RM sentences were made from the PJSENT2 sentences, one for the development test and one for the evaluation test. Distributing these over the (12x100) 1200 productions available in each gives 2 utterances per sentence. Simarly, two random selections of 150 words each were made from the pool of 600 spell-mode words, for development and evaluation tests. Distributing these over (12x50) 600 readings available yields 4 subject productions for each word.

3. SENTENCE IDENTIFICATION

Each sentence that was read has an identifying name. This sentence i.d. appears as a sub-field in the name of speech files holding recordings of the associated sentence. Table i is a key showing the significance of the different sentence i.d.'s.

4. LEXICON

In order to have a uniform and repeatable scoring, there is a need to specify each of the RM sentences in terms of a string of recognition units from a standard lexicon. It seems best to derive these representations from the prompts actually used in the collection of the data base instead of some other phase of the language model, since they are the most sure representation of what was probably said.

Dave Pallett (of NBS), who is organizing the scoring procedures, after soliciting and considering the opinions of interested parties, issued a memo giving rules for converting our prompts from normal orthography into strings of these lexical units. We call such representations SNOR's, for standard Normalized Orthographic Representations. In this kind of representation, the lexical units (or "words") are strings of non-blank characters separated by a blank. We wrote a set of lexicalizing rules in a quasi-linguistic format implementing Dave's rules but making explicit choices where there was some vagueness in his formulation. These rules are presented below as Figure 3.

The format of the rules is straightforward. In the symbol-defining section, certain variables are defined that range over specified strings of characters, and are used in the later definition of rules. In the rule-defining section, a list of rules for transforming character strings is given, of the form:

```
[A] --> [B] / [C] __ [D]; "comments"
```

The algorithm for rule application is simple. The rules apply to map an input buffer of characters into an output buffer of characters; the input, left environment, and right environment fields of each rule match to the input buffer, and if the rule applies, the output field of the rule is added to the output buffer. A cursor is initiated to point to the first character in the input buffer; at each cycle, the list of rules is searched from top down until either a rule is found that applies of the end of the list of rules is reached. If a rule is found that applies, the output of the rule is added to the output buffer and the input buffer cursor is advanced beyond the part of the input buffer that was matched by the input field of the rule. If no rule applies, the single character that the input buffer cursor points to is copied into the output buffer and the input buffer cursor is advanced by one.

```
C FILE TTPSNOR2.RLS
C SET OF RULES TO CONVERT T.I. PROMPT SENTENCES INTO
C STANDARD NORMALIZED ORTHOGRAPHIC REPRESENTATION (SNOR) FORMAT.
* INPUT : PCODEFILE=UD:[SPEECH.PH]CPASCII.DAT
* OUTPUT: PCODEFILE=UD:[SPEECH.PH]CPASCII.DAT
C RULE FAILURE ACTION = 'PASS'
*       ***** SYMBOL SECTION BEGINS HERE *****
 # = ' '/','/'.'/'?'/'!'/';'/':'/'"'"/'"'
 {AN+} = ' '/','/'.'/'?'/'!'/';'/':'/'-'
 $ANUM =  'T'/'N'/'R'/'S'/'H'/'D'/'L'/'F'/'C'/'M'/'G'/'P'/'W'/
+        'B'/'V'/'K'/'X'/'J'/'Q'/'Z'/'E'/'A'/'O'/'I'/'U'/'Y'/
+         '1'/'2'/'3'/'4'/'5'/'6'/'7'/'8'/'9'/'0'/'ZERO'
*       ***** RULE SECTION BEGINS HERE *****
* RULE FORMAT B
 [S'] => [S+S] / __ [#] ; weird possessive plural formation rule
 ['] => [+] ; "apostrophes become pluses" (for exception see TTPSNOR1.RLS)
 [.] => []  ; "abbreviations become single words, no end-of-sentence
C             punctuation"
 [?] => []  ;  "no end-of-sentence punctuation"
 [   ] => [ ]; delete multiple blanks
 [ ]  => [ ]; delete multiple blanks
C special hyphenated idioms:
 [ ] => [-] / [#DIEGO]  __  [GARCIA]
 [ ] => [-] / [#HONG]  __ [KONG]
 [ ] => [-] / [#ICE]  __ [NINE]
 [ ] => [-] / [#LAT]  __ [LONG]
 [ ] => [-] / [#NEW]  __ [YORK]
 [ ] => [-] / [#NEW]  __ [ZEALAND]
 [ ] => [-] / [#PAC]  __ [ALERT]
 [ ] => [-] / [#SAN]  __ [DIEGO]
 [ ] => [-] / [#SAN]  __ [FRAN]
C correct spelling of O in some alphanumeric strings:
 [ZERO] => [O] / [-] __
C supply weird spellings for some acronyms
 [CROVEL] => [CROVL] / [#] __
 [PACK] => [PAC]     / [#] __
 [TACKIN] => [TACAN] / [#] __
 [TASSEM] => [TASM]  / [#] __
 [-FLEET] => [FLT] / [LANT] __
 [-FLEET] => [FLT] / [PACK] __
C alphanumeric strings spelled without hyphens:
 [-] => [] / [{AN+}$ANUM]    [$ANUM{AN+}]
 [-] => [] / [.]  __  [$ANUM{AN+}] ; "D.D.D.-2-4-3"
C NOTE: THE RIGHT ENVIRONMENT IS NEEDED IN THE ABOVE RULE TO PREVENT
C IT FROM APPLYING TO, FOR INSTANCE, "S.Q.Q.-23", WHICH SHOULD BE "SQQ-23",
C NOT "SQQ23".
```

Figure 3. Major Rules for Lexicalizing Prompts.

The rules shown in Figure 3 are preceded by passes of rules, not shown, which capitalize all letters, delete the apostrophe in such abbreviated dates as "'87", and convert numeral strings to English words.

In brief, the rules: 1. eliminate punctuation; 2. convert letters to all capitals; 3. replace apostrophe with "+"; 4. combine certain "words" into single lexemes, using hyphens; 5. split up certain alphanumeric strings by inserting blanks.

TI has run a program to apply these rules to all prompts used in this data base and can supply the resulting SNOR strings and an alphabetical listing of the SNOR lexicon of these sentences to any interested parties.

5. RECORDING CONDITIONS

Recording conditions were nearly the same as has been reported at earlier DARPA workshops; in brief: subjects were seated in a sound-isolated recording booth; the director placed a Sennheiser SN 414 headset microphone on the subject and, using a template, positioned a B&K pressure microphone about 30 centimeters away from the subject's mouth, 20 degrees to the left; the subject was instructed to read the prompts appearing on a CRT screen in a "natural" voice; and speech was digitized directly onto disk at 20 ksps per channel. The automatic recording software system STEROIDS was used. Each recording was listened to by both the recording director and the subject to check for errors.

Before this data base recording began, the sound booth was retrofitted with a steel I-beam subfloor and air spring suspension system which reduced low-frequency (<100 Hz.) noise by about 20-25 dB.

The raw recordings were split into separate files for each channel, filtered, and down-sampled to 16 ksps as before and these versions of the speech were sent along with the original 20 ksps 2-channel files to NBS.

6. ERRATA

After a large amount of speech had been sent to NBS, they discovered that some recordings apparently had final words clipped off; this problem was called the "zero-tail" problem. Some investigation determined that the original recording was all right and that the problem was caused by a bug in general speech file software that was introduced with a program change in October. It was a "magic number" problem; only about 2% of the recordings made with the buggy software were affected. Recordings that had not been yet shipped by the time the bug was fixed were corrected before shipping. The recordings that had already been shipped were handled in a different manner: two "errata" tapes have been prepared, which contain corrected versions of already-shipped files that were found to have zero tails. These errata tapes are delivered with the data base, clearly marked, and users should make sure that the files on the errata tapes are used in place of the corresponding files with matching name.

An Architecture for Multiple Knowledge Sources

by James K. Baker
Dragon Systems, Inc.
90 Bridge St.
Newton, MA 02158
(617) 965-5200

Introduction

To achieve high performance continuous speech recognition, we need to bring to bear a wide variety of sources of knowledge. To achieve real-time continuous speech recognition, we must implement these knowledge sources working cooperatively in a very high speed computing environment, probably with many processors running in parallel. This paper discusses one approach to achieving these goals at a reasonable cost in the environment of a single workstation. It is furthermore a prime goal of Dragon Systems' project to provide an environment in which knowledge sources of many different types may be implemented.

The architecture, as seen by the knowledge source software, should be capable of mixing stochastic knowledge sources with deterministic knowledge sources. It should be capable of combining rule-based knowledge representations with pattern-matching based knowledge. It should provide for both parametric and non-parametric statistical procedures. Finally, it should facilitate the independent development of separate knowledge sources, possibly at remote sites.

Of course, Dragon Systems is not alone in working towards these overall goals. It is not claimed that we are anywhere close to a complete solution to the problem of many different knowledge sources cooperating in a real-time environment. Rather, the opposite is more nearly true--research on different knowledge sources cooperating in a real-time environment is likely to be of value specifically because of the current fragmentary state of our knowledge. Getting different kinds of knowledge sources to work together is still very much an exploratory activity. In this light, the current project is not trying to find an optimal multi-procesor, multi-knowledge source architecture, but merely an adequate one.

An important objective of this project is to develop techniques which apply not only to a substantial variety of the algorithms that we know today, but also to the algorithms that we might invent in the future. Therefore, both the hardware architecture and the software architecture must be general purpose, not tailored to a particular class of algorithm.

Hardware Architecture

The software architecture, discussed in more detail below, is specifically designed to be compatible with many of the existing parallel processor architectures. To meet the goal of fitting in a single workstation at moderate cost, however, the architecture shown in figure 1 has been chosen. The general framework is a simple tree: the host processor in the workstation itself with several clusters of processors, with each cluster implemented on one or two boards that plug into the peripheral bus of the workstation. Each cluster has a local system bus with a memory that is shared by the host and all the processors in the cluster. Each processor in the cluster also has a substantial amount of local memory of its own. The "knowledge" of the individual knowledge sources is stored in these local memories.

This architecture is not intended as a great innovation, similar architectures have been done before. Rather, it is a simple and reliable means of fitting a large amount of general purpose computation in a small space. With the multi-processor board that Dragon has designed, it is possible to fit up to 7 general purpose 2 MIP processors in a single slot of a personal computer. Over 50 MIPS could be available in a workstation. Even more computational power is feasible with more specialized processors.

In any multi-processor, multi-knowledge-source architecture a prime consideration is the communication between the knowledge sources and the communication between the processors. As will be discussed later, the software architecture that Dragon has adopted provides for a flexible, but very structured and controlled communication between the knowledge sources. The principal strategy which is used to reduce the amount of communication between processors, is to have each processor have sufficient processing power and a sufficient amount of local memory to implement one or more complete knowledge sources.

Choosing an architecture in which knowledge sources and processors are somewhat loosely coupled leads to different issues and different research questions than a more tightly coupled architecture. Thus code vectorization or converting scalar code to parallel code, which might be critical in a tightly coupled architecture, are insignificant in this architecture. On the other hand, partitioning the knowledge into separate local memories, which is unnecessary in an architecture in which every processor has immediate or near-immediate access to every memory location, is a critical issue in this hierarchical architecture. However, since it is a prime concern of this research project to study how knowledge sources of different types can work together, it is entirely appropriate to choose a hardware architecture in which the most important implementation issues occur at a similar level to the important issues in the functional architecture.

Note also that with each knowledge source located on a single general purpose processor, it is practical to do much of the development work for a knowledge source independently in a stand-alone environment, and still easily link the knowledge source into the rest of the system.

The specific implementation that Dragon Systems has designed uses up to seven 80286 processors on a mother-daughter board combination in a single slot in a high-end MS-DOS personal computer. There is 876K of local memory for each processor and also 64K of memory shared by all processors on the board and also by the host processor in the personal computer. The local system bus is essentially a standard Multibus restricted to the local board. Dragon's multi-processor board and its interface to the host CPU is described in much greater detail in the separate document "Multi-Processor Board."

Code written for this design should be upward portable to a design using 80386 processors with no recoding at all. It should be portable to workstations using other peripheral buses (Multibus, VME-bus or Unibus) and other operating systems (UNIX or VMS) with only moderate redesign and recoding. All of the knowledge source code, in particular, runs independently (within the specifications of the software architecture) on a single processor. An individual knowledge source is implemented independently of the higher-level hardware structure.

Since MS-DOS is not multi-tasking and not re-entrant, we have implemented a multi-tasking monitor to handle the low-level communication and synchronization between the processors and to simulate multi-processors on a single processor. Detailed specifications for these routines are available, but they will not be discussed further in this paper.

It also should be pointed out that although the multi-processor design has been completed, only a one-processor prototype has been constructed so far. Also, the benchmark software development has been done in a personal computer environment with limited memory, so even though the software architecture is intended for a multi-processor, multi-tasking environment, the current implementation runs non-real-time on a single processor without simulating the low-level communication details.

Software Architecture

The overall architecture of the multi-knowledge source system requires a clear distinction between the concepts of "knowledge" and of "data." "Knowledge" should be thought of as permanent information, such as properties of speech or facts of linguistics. "Data" is the information that has been computed about a particular utterance. "Data" is passed around among the knowledge sources and is used in the recognition process, but unless it is converted to "knowledge," it is not permanently stored. "Knowledge," on the other hand, is not shared. All knowledge is local to a particular knowledge source.

It is important to notice that these definitions are not merely definitions to distinguish the two kinds of information. Splitting all information into these two categories deliberately imposes very significant limitations on the overall system. "Knowledge" cannot be shared among knowledge sources; "data" cannot be saved permanently. Although some exceptions are allowed for efficiency, the distinction between "knowledge" and "data" is deliberately enforced to enhance the modularity of the knowledge sources.

For example, if two different knowledge sources both need models for the "expected" formant frequencies of each steady-state vowel (say one that is recognizing consonants from the formant transistions in adjacent vowels, and one that is recognizing the vowels themselves), then they should each have their own local copy of that knowledge. They can share the "data" about the formant frequencies estimated during a particular utterance, but they must each have their own local copy of the permanent "knowledge." If they each have their own local models, each knowledge source would still work if the other knowledge source were replaced by another knowledge source that used a completely different modeling method.

A knowledge source cannot directly call a function in another knowledge source. All communication is done by posting data on the "bulletin board." The system enforces a very strong degree of modularity on the knowledge sources.

In these restrictions, however, training is logically separated from recognition. Training, if necessary, can run offline using data that has temporarily been saved to files. Related knowledge sources that might be executing on separate processors at recognition time can be put on a single processor and can make direct calls to functions in other knowledge source modules. This mechanism should be used sparingly and not abused, but it is open-ended enough to allow any training algorithm implementation that is consistent with good structured program practice. Training does not have to follow the stricter discipline that is necessary for real-time computation on parallel processors.

How to do global training in a system that has many different kinds of knowledge sources is a very complex and intriguing question. In particular, it is an open research question as to how to combine knowledge sources that use fully automatic training with knowledge sources in which the training process involves interaction with a human expert. However, our investigations are still at a very preliminary stage. So, even though dicussion of this issue would be very welcome, it is not covered in this paper.

The discusion will now focus, therefore, on the loading of knowledge and the communication of data at recognition time. Five functions are specified for the communication of knowledge and data: three entry points that the knowledge source provides to the system (ks_load, ks_call, ks_unload, where "ks" would be replaced by a unique character string identifying the particular knowledge source) and two system functions (load_get_knowledge and post_get_data) that the knowledge source calls to get the actual knowledge or data.

The parameters and calling specifications for these functions is given in the Appendix.

From the point of view of an individual knowledge source, activity is divided into three parts: 1) loading and initializing the knowledge source, 2) the actual processing of utterances, and 3) cleaning up, freeing memory and unloading. Loading and unloading are mainly used in experiments in which not all of the knowledge will fit in available memory. In the multi-processor configuration, with a sufficient number of processors, all knowledge sources will be loaded at system initialization and would not need to be unloaded. The discussion, therefore will focus on ks_call and the processing of utterances to be recognized.

When a knowledge source is called, the only input parameter is "time." The parameter "p_post_list" is a pointer that the knowledge source sets to point to the first item in a list of items to be posted on the bulletin board, and "p_done_time" is an output value that the knowledge source sets to tell the system that it is finished up through the indicated time. The central data structure is called a "bulletin board" rather than a "blackboard," because previously posted items cannot be modified.

The knowledge source gets its actual input data by calling the function "post_get_data." This "input data on demand," lets an individual knowledge source request only the data that it needs rather than all the data the system has available, reducing the inter-processor communication. The system keeps track of the data sources that provide input to a given knowledge source. The system does not issue the "ks_call" for a particular value of "time" until all the input data is ready. Thus the knowledge source knows that it can call "post_get_data" for any of its input data sources for any time up to and including the current value of "time."

It is immediately apparent that this manner of calling the knowledge sources imposes a timewise "left-to-right" order on the processing of each utterance. This is one of the compromises that has been made to keep the system as simple as possible in some ways in order to make it as flexible and general as possible in others. Note that, since "p_done_time" may lag behind "time," each knowledge source may internally create a look-ahead buffer of arbitrary duration. Although there are some potential knowledge sources which might be very inefficient given this constraint of timewise processing, it is a reasonable constraint for a real-time system. The main limitation imposed by this processing method is in the possible implementations of syntax control, semantics, and language modeling generally, since any context dependence of duration less than a couple words can be easily handled with a look-ahead buffer. There are at least some parsing and language modeling methods that can work well within this constraint, with a limited look-ahead.

For the input data, a strict "timewise" sequence is imposed on a knowledge source. Once a knowledge source has called "post_get_data" for a particular time, it cannot go back to any earlier time. If it wants to reuse an item, it must buffer it internally. The output is not as restricted, the knowledge source can post data for any "post_time" greater than any previous "p_done_time." The system is responsible for buffering the output of the knowledge source until other knowledge sources have used it.

The constraints of this functional architecture thus are as follows:

1) Knowledge is local, not shared
2) Data is temporary, not saved
3) Utterances are processed timewise

Implicit constraints include:

4) Each knowledge source should use only a small fraction of its computation time communicating data
5) Each knowledge source should operate in real-time on a single 2 MIP processor with 876K of local memory
6) To minimize response time, knowledge sources should be designed to use data as soon as possible after it becomes available.

The "fraction" in constraint (4) must be smaller (in the range 5-10%) for this architecture than for some architectures, to prevent the local system bus from becoming a bottleneck. However, communication between several small knowledge sources clustered on a single processor doesn't count against this constraint. Since Dragon Systems has demonstrated real-time large vocabulary, natural language isolated word recognition on a single 1 MIP processor, it is believed that constraint (5) is not too great a limitation on the complexity of an individual knowledge source.

A knowledge source that requires a lot of processing but that doesn't use too much memory can be easily be partitioned to run on more than one processor simply by making duplicate copies of the local "knowledge." The greatest design constraint is for knowledge sources that need more than 876K of local memory for program code plus their "knowledge." Such knowledge sources must be partitioned to run as separate knowledge sources on more than one processor, without the ability to share knowledge among the partitioned knowledge sources.

It is easy to see that the constraints are all very broad and not specific to the kinds of knowledge sources involved. In this framework a "knowledge source" is any module of subroutines that satisfy the necessary constraints to be local to a processor. The module need not deal with what would conventionally be called "knowledge." Thus an FFT routine would be a "knowledge source" as long it followed the calling conventions and received and sent all its data by posting on the bulletin board. The FFT routine would have an empty set of "knowledge," even in the formal sense, unless the coefficient table was loaded as "knowledge." Thus a "knowledge" source need not actually have any "knowledge."

On the other hand, a knowledge source could be very complex. It could be a complete rule-based phoneme recognizer or a complete hidden Markov model word recognizer. Some knowledge sources could be "translators" that would allow knowledge sources of different types to cooperate with each other. With this functional architecture it will be possible to run "plug-and-replace" experiments with several different versions of the knowledge source that does a particular task. The other knowledge sources will not need to be explicitly aware of which of the experimental knowledge sources is in the system.

As a matter of good programming practice, it is generally preferable to break the recognition task up into a larger number of simpler knowledge sources. Thus different knowledge sources might specialize on different phoneme classes rather than being combined into a single knowledge source. Each knowledge source should be only a few hundred lines of code in a higher level language.

Current work is proceeding on two fronts: within this software architecture Dragon is implementing a complete connected word recognizer as a feasiblity proof of the knowledge source partitioning and the timewise processing and as a platform for studying communication and performance bottlenecks. Dragon is also implementing "benchmark" versions of a variety of novel algorithms, e. g. neural networks, to see how they might be incorporated into this framework. Dragon also invites other DARPA sites to submit knowledge sources in either source code or object code form. The greater the variety of knowledge sources that can work together in a cooperative environment, the greater will be the benefit for the whole speech recognition research community.

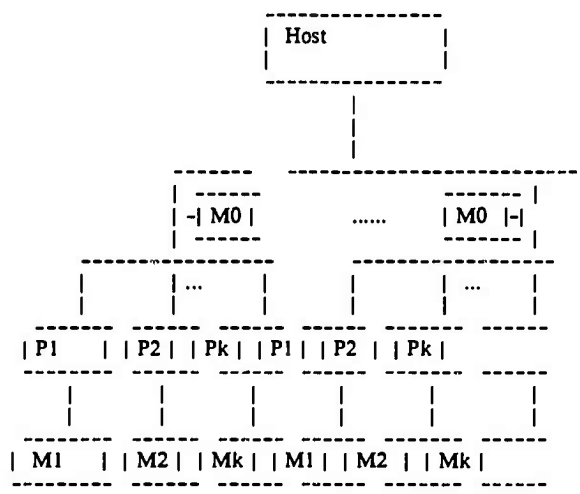


Figure 1

Appendix

```
BOOL ks_load(ks,data_size,data_handle)
        /* Returns YES if already loaded */
    uns16 ks;       /* The unique handle the system has assigned to
                       this knowledge source, used when calling
                       load_get_knowledge. */
*/
    int32 data_size;        /* The size in bytes of the block of
                       knowledge that the system has available to
                       pass to this knowledge source */
    int16 data_handle;      /* A handle that the system has assigned
                       that the knowledge source should use when calling
                       load_get_knowledge */

BOOL ks_call(time,p_post,p_done_time)
        /* Returns YES for end-of-data condition similar to EOF
*/
    uns16 time; /* Current time as measured by the system.  Being
                       called with this value of "time" tells this
                       knowledge source that any input data that it
                       all knowledge sources that send input data to
                       this knowledge source have reported to the
                       system that they are done posting up through this
                       value of time. */
    struct POST_LIST **p_post;    /* A pointer to a list of items
                       to be posted on the "bulletin board" */
    uns16 *p_done_time; /* The knowledge source tells the system

                    that it has finished posting all items to be
                    posted at time up to and including p_done_time.
*/
```

```
struct POST_LIST {
     uns16 size; /* actual size of the data in this item */
     uns16 post_time;  /* the time slot on the bulletin board
          at which this item is to be posted */
     struct POST_LIST *next_post;  /* pointer to the next item
          to be posted, if any */
     char data[MAX_DATA_SIZE];
                                   /* The actual data, which may
          have any kind of information, but whose internal
          structure is only known to the knowledge sources that
          use it. */
};

BOOL ks_unload()

int32 load_get_knowledge(ks,data_size,data_area,data_handle)
          /* Return the actual number of bytes sent */
     uns16 ks;  /* Knowledge source handle */
     int32 data_size;         /* Size of buffer area in which to
          put a portion of the knowledge. */
     char *data_area;         /* Pointer to buffer area */
     int16 data_handle;       /* A handle for the knowledge data_area,
          comparable to a file pointer. */
          /* (In the current implementation load_get_knowledge is
                    functionally similar to a
                    read(data_handle,data_area,data_size) ) */

BOOL post_get_data(handle,post_time,ship_to,ship_size,remain,
     complete,eof)
          /* Returns YES if there more to come for the current
post_time */
     uns16 handle; /* Handle identifying the the data source */
     uns16 post_time; /* The posting time for which data is
          requested.  Any knowledge source for which ks_call has
          called with a particular value of "time", may call
          post_get_data if any value of post_time<=time. */
     char *ship_to;           /* Buffer in which to put a block of
          the data. */
     uns16 *ship_size;        /* On input: the size of the ship_to
                              buffer
                              On output: the actual number of bytes
                              sent                              */
     uns16 *remain;           /* Number of bytes remaining in the
          data that was posted for the current time. */
     BOOL *complete;          /* This buffer includes the end of a
          complete item */
     BOOL *eof;  /* YES if there is no more data (for this or
          any greater value of post_time */
```

Experiments in Isolated Digit Recognition with a Cochlear Model—An Update

Richard F. Lyon and Eric P. Loeb

Schlumberger Palo Alto Research
3340 Hillview Avenue
Palo Alto, CA 94304

Abstract

We have conducted speaker-independent isolated digit recognition experiments using vector quantized cochleagrams. Without the use of time order information, we were able to achieve a recognition rate of 98.3%. With a modified Viterbi algorithm we achieved a rate of 99.1%, on a test with a larger talker population. Since these accuracies are not far apart, we must call into question the effectiveness with which the Viterbi algorithm uses time order information. These results demonstrate that the auditory sprectrum approach leads to high performance even with simple non-parametric techniques and phoneme-level word models. The results presented here update the results presented at ICASSP 87 [Loeb et al. '87]; they verify the prediction that accuracy would be significantly improved by doubling the training and testing talker populations, and by using two repetitions of each digit from each talker in training.

1 Introduction

We have conducted a number of isolated digit recognition experiments in an effort to evaluate the potential of an auditory model front end. The experiments emphasize non-parametric approaches and techniques that use little or no time order information. We wished to set high performance standards for future experiments while estimating the relative importance of the various sources of information in the data.

Although the front end for our experiments is a cochlear model [Lyon '82], there is nothing explicitly neural about our techniques. They could be applied to any other vector quantized representation. Many of the experiments are interesting as techniques for the use of non-parametric statistics in spite of the shortage of training data. Since every experiment in the first group, originally presented at ICASSP 87 [Loeb et al. '87], uses the same training and testing data sets, those results are directly comparable; later experiments extend some results to larger training and testing sets.

The bulk of this paper describes the various recognition methods that we tested. We start with the non-time-order methods. These include a "Basic Method", four modifications to it, and a section on vector quantization methods. After that we describe our two time-order methods. Just before we conclude, we interpret some of our methods and results in terms of neural networks.

2 General Methods

The first repetitions of the isolated digits in the training subset of the TI Connected Digit Database (sampling rate 20kHz) were analyzed by our cochlear model. The model produces a discrete-time 92-channel spectrum, which is down-sampled to 1 kHz and quantized by a standard Euclidean quantizer with 1024 codewords. The quantizer codebook was trained on the first repetitions of all the training speakers using the standard K-means algorithm.

In the first group of experiments to be described, half of the 112 training speakers were used for training and the other half were used for testing. Thus the recognition results in these experiments are ostensibly speaker-independent. We cannot claim total speaker-independence because we used both sets of speakers to build our vector quantizer. Since this caused two of the codewords never to occur in our training set, the net result is actually poorer performance than we find in extending to more speakers.

In later experiments, up to four times as much training data was used, by using all 112 training speakers, with two repetitions from each. One repetition of each digit from each testing speaker has been analyzed so far.

3 Definitions

codeword an integer in [0, B] where B $\leq$ 1023. Each codeword corresponds to some subset of $\Re^{92}$, and the set of codewords corresponds to a partition of $\Re^{92}$.

utterance the sequence of codewords derived from the cochleagram of one of the speakers saying one of the vocabulary words.

utterance histogram a vector $\vec{H}(A)$, where $\vec{H}_{cw}(A)$ is the number of occurrences of codeword cw in utterance A.

guess the index to a vocabulary word. A guess is the result of some recognition method operating on a test utterance.

guess vector a <vocabulary-size>-dimensional vector of word log probabilities. If a recognition method generates a guess vector, then it will always output the index of the most probable word as its guess.

4 The Basic M

A matrix of condit ilities of observations (codewords) given the word s first generated. Probabilities are estimated from a count of the number of occurrences of each codeword within all training utterances of each vocabulary word. Then, for recognition, each word in the vocabulary is scored by adding the log likelihoods for all time samples in the unknown utterance; since these scores do not depend on the order of occurrence of the samples, they are most easily computed by multiplying the log probability matrix by the utterance histogram.

Now each codeword, cw, indexes a vector $\vec{V}(cw)$ in our matrix, whose i^{th} component is given by

$$\vec{V}_i(cw) = -\log \Pr[\text{codeword } cw \mid \text{word } i].$$

In this notation the same guess vector can be formed by accumulating the $\vec{V}(cw)$'s indexed by each codeword found in sequence in the test utterance.

This simple program gives 94.97% correct recognition on the first repetitions (see Table 1), and 95.54% on both repetitions (see Table 2). This implies that the codewords (and thus the underlying cochleagrams) are doing a good job of acoustically separating our vocabulary words.

It is interesting to note that an earlier version of the codebook, in which the K-means algorithm had not iterated to convergence, gave 94.16% recognition on the first reps. This indicates that the method of codebook vector production is an important component of a quantizer-based system.

Finally, when we used the basic method on quantized LPC spectra we achieved 88.31% recognition. The LPC codebook had 1024 codewords made by the K-means algorithm, but the LPC quantizer produced one codeword every 10 msec. When we downsampled the cochlear quantizer to the same rate the basic method gave 93.83% recognition. We repeated this test with eight different training and testing populations. In all cases the number of LPC errors were roughly double the number of cochlear errors.

5 Simple Variations on the Basic Method

5.1 Codeword Grouping – OR type

Let us suppose there are several codewords covering the spectra produced by /s/ sounds. Then the majority of the observations of these codewords will occur in the vocabulary words containing /s/ sounds. In the task at hand these words are *six* and *seven*. So, if we assign one new number to every codeword, cw, such that most of the observations of cw occur in the words *six* and *seven*, then this new number should be a good indicator of the /s/ sound.

To implement this idea we need a parameter *coverage-proportion*. We map each codeword, cw, to a list of the vocabulary words that account for at least *coverage-proportion* of the observations of cw. Next we map these lists to integers (i.e., we number them). The composition of the two mappings is a many to one map from the original codewords to some new codewords. We then use the new codewords in the basic method.

There are two important variations in the mapping from lists to numbers. We make an unordered grouping by numbering the lists as sets so that the order of the words does not matter. We make an ordered grouping by numbering the lists so that the order of the words does matter.

The results are on the "Basic" row of Table 1 and Table 2. In Table 1 the groupings are assumed to be unordered (number as sets). In Table 2 we compare the unordered grouping with *coverage-proportion* = 0.80 to the ordered grouping (number as lists) with the same *coverage-proportion*.

Although this method is of marginal use in improving performance, it does reduce the number of codewords. In Table 1 the 0.95 (unordered) grouping reduces the number of codewords from the original 1024 to 405, and the 0.80 (unordered) grouping has 313 codewords. In Table 2 the 0.80 unordered grouping has 308 codewords. This is different from the same grouping in Table 1 because we estimated the codeword distributions with both repetitions. The 0.80 ordered grouping in Table 2 has 462 codewords.

Some of the sets/lists that we use to make these groupings have very clear phonetic content. For example the 0.80 unordered grouping of Table 1 maps 71 of the original codewords to the set {*six* , *seven*}, 39 originals to the set {*three* , *zero*}, and 22 to {*one* , *nine*}. Although many of the sets do not have such obvious phonetic content, most of the sets that represent large numbers of original codewords do. Thus we may have found a way to generate phonetically meaningful labels without imposing our pre-conceptions upon the data. In the future, we hope to extend this method to the grouping of sequences of codewords.

5.2 Histogram Compression

Without changing the training procedure, we map the histogram of each testing utterance to its log. I.E If codeword X occurs N times in testing utterance A, then the contribution of codeword X to our guess vector for utterance A will be the product of log (1 + N) and row number X of the matrix of log probabilities.

Notice that we can achieve approximate log compression without the use of histograms by letting the n^{th} response to codeword X be 1/n. This is highly reminiscent of habituation. Thus the histogram compression method is both neurally plausible and extendable to continuous speech recognition.

The results on the first two lines of the result tables suggest that more codewords are better for this method. We should expect as much, since compression gives more equal weight to each of the codewords than the basic method. We suspect that compression offers greater improvements than any of the codeword groupings because it does a better job of reducing the effects of uninformative codewords.

When compression and time splitting (Section 7.1) are used together we begin to rival our best Viterbi algorithm results. This combined method is very simple and uses virtually no time information at all. This suggests the possibility that our Viterbi algorithm might be improved by the addition of a compressive operation.

5.3 Non-Occurring Codewords (or Necessity)

The basic method will often guess "zero" when the input is an "oh". The reason is that the method is one of sufficiency – it has no way of necessitating a /z/ sound before guessing "zero". Thus we need to modify the basic method to make use of the codewords that do not occur.

For each codeword, *cw*, that does not occur in the test utterance we add an inverse of $\vec{V}(cw)$ (Section 4) to our guess vector. The inverse of $\vec{V}(cw)$ was computed by subtracting each element, $\vec{V}_i(cw)$, from the maximum element of $\vec{V}(cw)$.

If we examine the differences between the "Basic" and "Necessity" results in Table 1, then it appears that this method becomes more successful as the number of codewords decreases. If this were simply the case, however, we would expect an even larger improvement in a 0.50 unordered grouping, which had only 156 codewords. The recognition rates with this grouping were 90.26% with the basic method and 91.56% with the necessity method. So the utility of the necessity method seems to depend on the extent to which our codewords correspond to phonetic units.

5.4 Several Ranges for Each Codeword

The basic method has difficulty distinguishing between "nine" and "one". We would expect "nine" to have roughly twice as many milliseconds of /n/ as "one", but the basic method can not take advantage of this. We need the probability vectors $\vec{V}(cw)$ (Section 4) to be functions of the number of occurrences of *cw* as well as of *cw*.

To do this, we used the basic method with 6 matrices. If we let N be the number of occurrences of codeword X in utterance A, then utterance A will contribute to or use the $\lfloor \log(1+2N) \rfloor$th matrix for codeword X.

Each of our non-time-order methods is a function that maps every point in the space of utterance histograms to a log distribution. The improvements shown in Table 1 may be due to our adding more detail to this function. Examination of Table 2, however, shows that the ranges method works about as well as the compression method (Section 5.2), so the this method may work because it allows each codeword to contribute equally to the final decision. Alternatively, the improvements may be due to the fact that we are now taking into account the average number of occurrences of a codeword among those utterances in which it occurs. This statistic represents durational information.

5.5 Codeword Groupings – AND type

The codewords are not independent. It is therefore interesting to consider higher order conditionals such as the probability of word W given codeword X, codeword Y, and no codeword Z. The number of conditionals of this form is, however, prohibitively huge. We thus have no choice but to concentrate on groups that have a reasonable chance of occuring.

We mapped each *ranged* codeword, RCW, to a list of codewords that had an above-threshold correlation to RCW in the training data. We then considered each of these lists to be a codeword, where the number of occuraces of a list is the geometric mean of the number of occurances of each of its elements. We applied the basic method (Section 4) to the unique multi-element lists and added the resulting guess vector to the guess vector obtained from the ranges method (Section 5.4).

By adding the guess vectors of this method to the guess vectors from the (compressed, original codewords) basic method we were able to get a recognition rate of 98.3%. Since this result was totally ad hoc, it seems reasonable to expect that a carefully built system can achieve 99% recognition with no time order information at all.

These AND-groupings took a great deal of time and memory to implement. We consider these groups to be the equivalent of word models. Since this method produced a significant improvement we expect we will be able to find a simpler, more effective means of using inter-codeword correlations to generate useful word models.

6 Vector Quantization Methods

Let us suppose we have a speech recognizer box. Its input is a speech waveform or sequence of observations, which may be thought of as a vector. Its output is a word, which may be thought of as a scalar. Thus our speech recognizer is, in fact, a vector quantizer. Can it be implemented directly as one?

To cut down the pattern space some, we use binary histograms (i.e., each codeword either occurs or does not occur in the test utterance) with Euclidean distance, and the standard K-means algorithm to construct a codebook. A test utterance then maps to its closest codebook vector, which in turn tells us which vocabulary word to guess (the one that most frequently mapped to that codebook vector in training).

In a second experiment codebook vector k was set to be the centroid of all the training vectors for vocabulary word k. In another experiment we formed 32 ortho-normal vectors from the 32 codebook vectors of the first experiment. We then used the basic method by finding the projection of the test utterance on each of these vectors and summing the product of these numbers and the appropriate log probability vectors.

In the first experiment, a codebook of size 16 gave recognition = 66.6%. When size = 32, recognition = 76.5%, and when size = 128, recognition = 80.5%.

In the second experiment, with one codeword per vocabulary word, we got 92.69% recognition. This gives a rough idea of the efficacy of the K-means algorithm in approximating the "correct" decision boundaries.

The third experiment gave recognition = 70.13%. Since the codebook vectors we ortho-normalized for this experiment were the same 32 vectors used in the 32 vector part of the first experiment, it is clear that this method was of no help whatsoever.

7 Time Order Methods

7.1 Time Splitting

It is surprising that we can do so well without any time information, but we will need to use it eventually. As a simple extension of the methods we have tried so far, we use the basic method on the first and second time-halves of the utterances. Thus each test utterance will produce two guess vectors : one for its beginning and one for its end. The final guess vector will be the sum of these. In a second experiment, we use the necessity look-up method (Section 5.3) on both time-halves. In a third experiment, we use the range method (Section 5.4) on both time-halves. Finally, we use the compression method (Section 5.2) on both time halves.

As in Section 5.2, the fact that such a simple method could provide so much of an improvement (see Table 1), confirms that the time order information will be extremely helpful when used properly. The results of the combined time splitting and codeword ranges methods are respectable, but they depend far too heavily on the grouping parameter to be considered useful.

7.2 Viterbi Algorithm

The Viterbi algorithm is well known in speech recognition. We have applied it using simple finite-state word models similar to those used by Bush and Kopec [Bush '85].

The cost metric used by the Viterbi algorithm in finding a best model-based segmentation is $-\log \Pr[\text{codeword} \mid \text{state}]$, as in our basic method. The state tables were initially trained using segmentations found by Bush and Kopec's LPC-based recognizer; they have been retrained and modified to improve performance. In comparing the fits of the various word models, we used measures other than total cost (probability), as described elsewhere [Lyon '87]. In particular, the average costs in each state were given equal weight, rather than giving equal weight per unit of time; and a term was added to account for the probability of the duration in each state, after the best fit to each model was found.

The scores reported in Table 1 are the best of several variations. Other variations on the scoring function, for example using total Viterbi cost or omitting durational probabilities, resulted in up to twice as many errors. We were able to reduce the error rate from 1.62% to 0.91% using the same codebook but twice as many training repetitions (still testing only on first repetitions). Testing on both repetitions from the other 56 speakers increases the error rate to 1.46%; this doubling of training and testing data leads to error rates as low as 1.70% in the best of the non-time-order tests.

It is interesting that the finite-state models give a performance that is at best only slightly better than the techniques that use little or no time sequence information. Better techniques for handling timing and dynamics are clearly still needed.

In a later test, training talkers were grouped into two clusters based on codeword occurrence histograms accumulated across all of their utterances, and separate state models were trained on each cluster. It was found that the two clusters (found by K-means algorithm) partitioned the talkers almost perfectly into males and females. Recognition using both sets of models provided no significant difference from using a single set of models, contrary to our positive experience with separate male and female models in an LPC-based digit recognizer. This is probably due to the fact that the vocalic auditory spectra resolve harmonics enough to distinguish high and low pitches, so that the simple non-parametric techniques already separate male from female fairly well.

Finally, training on two repetitions from all 112 training talkers and testing on 113 new talkers (first reps only), the error rate is reduced to only 0.89% (11 errors in 1243). The second reps are expected to lead to up to twice as many errors, based on our experience with the 112 training talkers, so the net system performance is estimated at not worse than 33 errors in 2486 (which should be compared with TI's best published result of only 14 errors in 2486 tokens [Bochieri et al. '86]).

Recognition Method	OR-Grouping			Time Order
	Original	0.95	0.80	
Basic Method	94.97%	95.45%	95.29%	No
... with Compression	96.75%	96.10%	94.64%	No
... with Necessity	94.16%	95.62%	96.10%	No
... with Ranges	94.81%	97.24%	95.94%	No
Time Splitting	96.10%	95.94%	96.27%	Some
... with Necessity	94.81%	96.43%	97.56%	Some
... with Ranges	95.78%	98.05%	97.56%	Some
Viterbi Algorithm	98.38%	94.15%	—	Yes

Table 1: Recognition percentage for 616 test utterances. Grouping numbers are the *coverage-proportion* values from Section 5.1

Recognition Method	OR-Grouping		
	Original	0.80 Unordered	0.80 Ordered
Basic Method	95.54%	96.19%	96.02%
... with Compression	97.40%	96.02%	96.59%
... with Necessity	94.40%	96.83%	96.43%
... with Ranges	97.56%	—	97.48%
... with AND-Grouping	—	—	98.05%
Time Splitting	96.92%	96.92%	96.59%
... with Compression	98.38%	97.32%	97.73%
... with Ranges	97.65%	—	98.05%
Viterbi Algorithm	98.54%	—	—

Table 2: Recognition percentages for 1232 test utterances. Half of these utterance were used for Table 1. The other half are the second repetitions.

8 Neural Equivalents

The four variations to the basic method were produced by a simple neural modelling paradigm. Given a numerical system like the Basic Method we describe the system as a neural network, find some way to make this network more realistic, and then test the numerical system version of the more realistic neural network.

We consider the Basic Method (Section 4) to be equivalent to a neural network in which each codeword corresponds to an input neuron, and each vocabulary word corresponds to an output neuron. Under this equivalence, input neurons fires once each time their codewords occur in the test utterance. Input neuron cw is connected to output neuron i by a linear excitatory synapse of weight $\bar{V}_i(cw)$. Output neurons sum their inputs, and the number of the cell with the lowest value (most probable) is our guess.

The OR-groupings (Section 5.1) were inspired by a kind of connectionist model in which each input neuron excites the set of output neurons with which it is associated. Each output neuron then excites its input neurons. We believed that such a system would ultimately make indistiguishable the input neurons that belong to the same set of output neurons.

Two of the other variations were motivated by the fact that neurons are not simple linear devices. For the Compression method (Section 5.2) we considered that the response curves of r neurons look like bounded logarithms. We thought of the Ranges method (Section 5.4) when we considered that real neural pools tend to divide the numbers they encode into approximate log ranges [Brooks '86, Chapters 3 and 4]. Thus for each codeword we would expect one neuron to fire when the codeword does not occur, one to be sensitive to a small number of occurrences, another to fire in proportion to a larger range of occurrences, and another that only fires during large inputs.

9 Conclusions

It is clear that our cochlear model provides an adequate, if not superior, spectral representation. The unexpectedly good performance of the simple methods implies that the cochleagrams are effectively separating phonetic units. Furthermore, the cochleagrams appear to be better at separating phonetic units than LPC spectra.

The results of our other non-time-order experiments are intriguing. We now know that there exists a vector quantizer that maps utterance histograms to words with at least 98.3% accuracy (Section 5.5). Although our direct quantizer experiments are incomplete, it appears as though the K-means algorithm cannot reach this level of accuracy. If that is the case, then we can perhaps achieve a substantial improvement by using a better quantizer on the cochleagrams. By analogy with our recognition methods, we expect that we can improvement the K-means algorithm by assigning different weights to each dimension and by feeding back the distribution of each vocabulary word with respect to each codeword.

It is important to note that we have many results ranging from 9% to 90%. The methods reported here are methods that worked, and the methods that worked have generally been neurally motivated. At worst we have a way of thinking about the problem that leads to some solutions. At best, the brain uses the only possible solution, and we are using a non-random means of finding that solution.

The most difficult remaining question is how to handle time order information. The proximity of our Viterbi results to the results that used little or no time order information forces us to conclude either that time order information is not so useful as had been thought, or that the Viterbi algorithm with simple word models does not use it very effectively. If we suppose that the AND-grouping method (Section 5.5) uses most of the time information – namely, the codeword durations and the codeword co-occurances, then it seems that the actual order in which codewords occur is not terribly important. On the other hand, this is the only time information available to the time splitting method (Section 7.1), which was able to out-perform all methods except the Viterbi algorithm. Thus we cannot call any particular piece of information crucial. In this situation our best option is to go back and insure that each component of our system is exceptionally well done. For this reason we believe that the next set of experiments should involve different quantizations of the original cochleagrams.

References

[Bochieri et al. '86] Enrico L. Bochieri and George Doddington, "Speaker Independent Digit Recognition with Reference Frame-Specific Distance Measures," *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Tokyo, April 1986.

[Brooks '86] Vernon B. Brooks. *The Neural Basis of Motor Control*, Oxford University Press, Inc., New York, 1986.

[Bush '85] Marcia Bush and Gary Kopec, "Evaluation of a Network-Based Isolated Digit Recognizer Using the TI Multi-Dialect Database," *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Tampa, March 1985.

[Loeb et al. '87] Eric P. Loeb and Richard F. Lyon "Experiments in Isolated Digit Recognition with a Cochlear Model," *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Dallas, April 1987.

[Lyon '82] Richard F. Lyon, "A Computational Model of Filtering, Detection, and Compression in the Cochlea," *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Paris, May 1982.

[Lyon '87] Richard F. Lyon "Speech Recognition in Scale Space," *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Dallas, April 1987.

Research supported by DARPA contract #N00039-85-C-0585.

CONNECTED WORD RECOGNIZER ON A MULTIPROCESSOR SYSTEM

B.I. Pawate, M.L. McMahan, R.H. Wiggins,
G.R. Doddington, P.K. Rajasekaran
Speech & Image Understanding Laboratory
Computer Science Center
Texas Instruments, Inc.
Dallas, Texas 75265

ABSTRACT

Speech recognition algorithms employing a similarity measure between the input speech utterance and the stored reference patterns to determine recognition of a word/sentence are computationally intensive. The instantaneous vocabulary size that can be handled in real-time is relatively small. This limitation can be alleviated by either using multiple programmable processors or by using special purpose hardware to handle the computation-intensive tasks. In a research environment the former approach is preferred, because improvements to the algorithm can rapidly be incorporated and their effects studied in real-time. Texas Instruments has developed a multiple-processor architecture based on the TMS32020 DSP, called Odyssey, that interfaces with Explorer, a symbolic computer. This paper addresses the issues involved in partitioning and allocating tasks in a multiple-processor environment to maximise throughput, and discusses the implementation of a grammar-driven speaker-dependent connected-word recognizer (GDCWR) as an example application that uses the power of multiple processors.

Introduction

Many speech recognition algorithms extract a feature vector from the input signal at a rate of 25 to 50 times per second [1]. Vocabulary words may then be represented by sequences of feature vectors each representing the spectral content of the signal over a short period of time called a frame. In the recognition process, new feature vectors are computed at the frame rate(25 to 50 Hz) and compared to every reference vector in every vocabulary word. Comparision involves Euclidean distances between N-element vectors, where N is typically between 10 to 20, and dynamic programming to optimally time-align reference vectors with the input speech vector. This process is computationally demanding and limits the size of the active vocabulary that can be processed in real-time. One way to overcome this limitation is to use multiple programmable processors to distribute this loading. Texas Instruments has developed a multiple processor architecture called Odyssey. In a research environment, the multi-processor programmability is extremely desirable since such an architecture can be used as a protype to test and evaluate advanced robust speech recognition/DSP algorithms.

The Odyssey system [2] is an expandable, multiple digital signal processor (DSP) architecture based on the TMS32020 programmable microcomputer[3]. Key features of the board are: 20 million multiply/accumulates per second, 512K bytes of data space, and expandability to 16 boards on a NuBus host.

The Odyssey host is Texas Instruments' Explorer[4], a LISP machine workstation. Software has been provided which extends the high productivity environment of the Explorer into the area of digital signal processing. This provides an environment to perform many intelligent signal processing tasks by associating meaningful relationships between quantitative (signal processing) and qualitative (symbolic processing) entities to develop inferences using expert system technology. Applications such as grammar-driven connected-speech recognition, neural network simulation, and generation of speech with natural language generation techniques are some of the tasks that can utilize the computationl power of the mutiple DSP and symbolic processing.

Grammar-Driven Connected-Word Recognizer (GDCWR)

Figure 1 shows a block diagram of the GDCWR system [5]. An isolated recognizer outputs all the words that are hypothesized along with their corresponding distance scores and estimated durations. The basic technology of the word recognizer is a modification of the original Texas Instruments LPC-based isolated word recognition system[5]. The sentence hypothesizer constructs probable sentences from the word hypotheses and their time marks, and invokes grammatical constraints to consider only the admissable paths to output a recognized sentence with the lowest distance score. The list of possible subsentences is pruned to minimize both memory and processing requirements. The distance measure for the sentence has three parts: the first com-

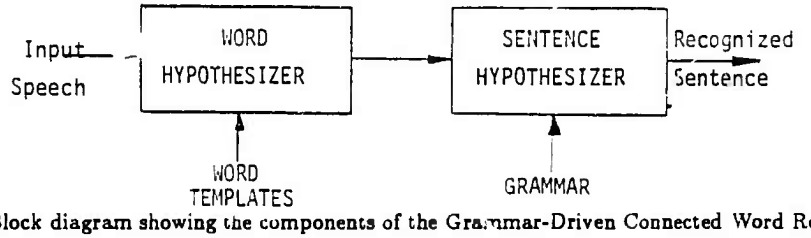


Figure 1: Block diagram showing the components of the Grammar-Driven Connected Word Recognizer.

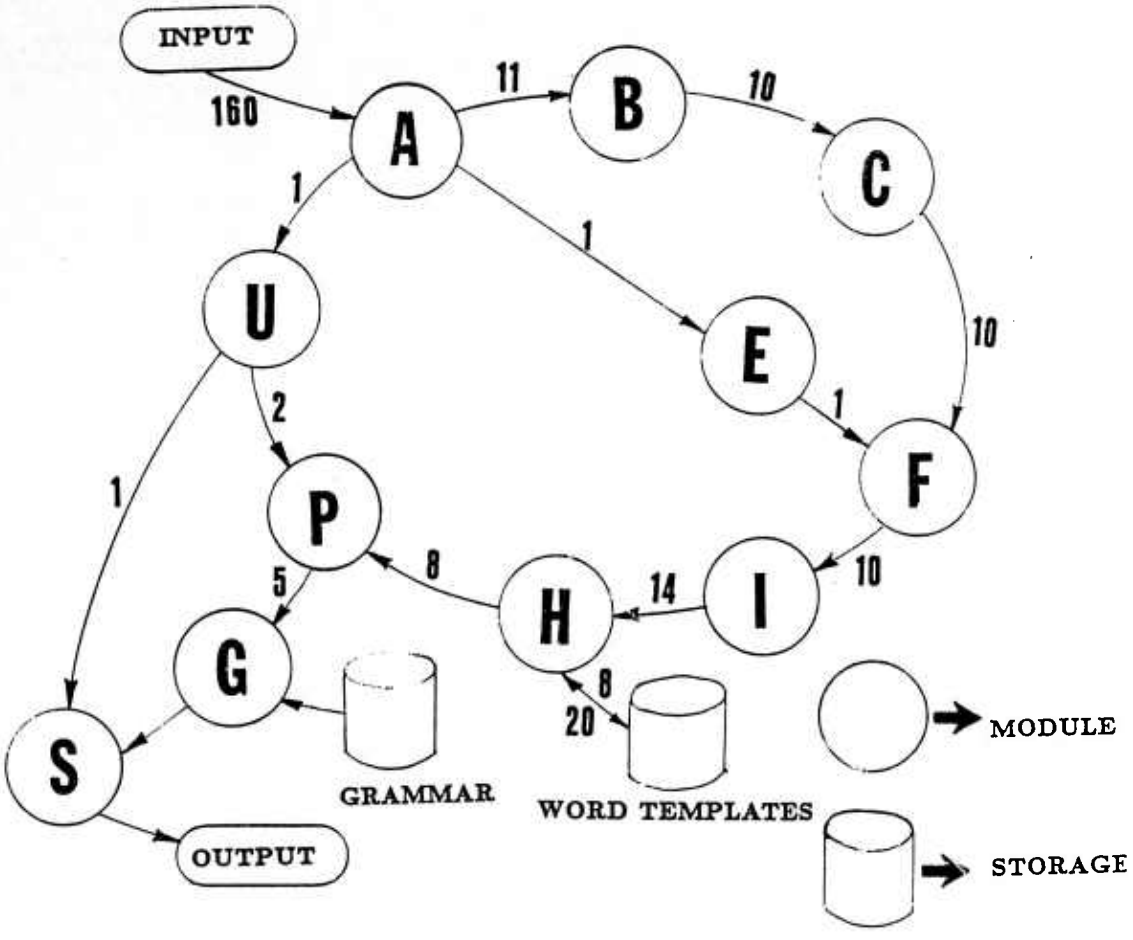


Figure 2: Task Partitioning

ponent is the sum of individual word distance scores multiplied by corresponding word durations; the second is a penalty for overlap or underlap of adjacent words; and the third is a silence (null speech) distance measure. An important feature of the recognizer is that the sentence hypothesizer does not control the isolated word recognizer by any feedback. This ensures that all information is preserved for late binding and possible recovery from higher level errors.

The loading of the similarity measurements is a fixed predictable function of the vocabulary size whereas the loading of the sentence hypothesizer is not fixed and is a function of the size and complexity of the grammar and the input utterance.

Task Partitioning and Allocation

In an ideal multiple processor environment one would expect the throughput of the system to increase linearly as the number of processors increases. However, this is not always true. In practice, the throughput in a multiple processor system increases significantly only for the first few additional processors and in fact begins to decrease after a certain number of processors [6]. This is due to increased interprocessor communication (IPC). This occurs when software modules, resident on different processors, need to communicate with each other. Communication protocols, management of storage, waiting time in queues etc. all contribute to the overhead. This overhead grows rapidly with large numbers of highly interacting processors and the system throughput actually begins to decrease. This is referred to as the *saturation* effect.

The designer is now faced with a dilemma. In order to exploit the computing resources offered by the multiple processor system he needs to balance the load. But balancing the load creates interprocessor overhead which needs to be kept as low as possible. One way to compromise these two conflicting factors is to allocate closely related software modules to the same processor and keep the communication between processors to a bare minimum. This demands a thorough understanding of the algorithm and the flow of data involved.

The first step is to partition the algorithm into several individual sub-tasks or modules. For example, GDCWR has been split up into several subroutines which have been arbitrarily named A, B, C, etc. Subroutine A calculates the 11 autocorrelation values from a frame of digitized speech samples, B is a routine that computes the reflection coeffecients and so on. These subroutines, represented by circles, are shown in Figure 2 and are connected in accordance with the flow of data. The number of words being passed from one sub-routine to another on a per frame basis represents the inter-module communication and has been placed on the connecting arcs. This process is known as task partitioning.

Once the task partitioning is completed, the next step is to allocate these modules to different processors so that the system throughput is maximized. This is known as task allocation. It is during this phase of the design that one has to balance the two conflicting factors of load distribution and minimum interprocessor communication. To maximize throughput, the individual processors should be able to run *autonomously* to the extent possible.

The first step in task allocation is to identify those routines that are closely related and/or communicate with one another extensively. In Figure 2 routines A, B, C, E , F and I are closely related and therefore fused together to form a bigger module called the Preprocessor which is allocated to one processor. It was found that H contributes to more than 50% of the loading and limits the vocabulary size. An entire processor must therefore be devoted to doing H. However, there is considerable traffic between H and P and interprocessor communication would be increased if these routines were resident on different processors. H and P are therefore fused together to form a bigger module called the Word Hypothesizer and allocated to the another processor. U is a routine that could be allocated to the Word Hypothesizer or the Preprocessor, but since we wish to allocate as much CPU time to the Word Hypothesizer as possible to do the similarity measurements, U is allocated to the Preprocessor. The remaining routines G and S comprise the Sentence Recognizer and are allocated to another processor. This completes the task allocation of the CWR software. The basic recognition system therefore requires three processors viz., the Preprocessor, the Word Hypothesizer and the Sentence Hypothesizer. The parallelism offered by a multiprocessor architecture can now be utilized to increase the active vocabulary size by the concurrent execution of the Word Hypothesizer on two or more processors with each processor addressing a smaller subset of the vocabulary.

Figure 3 shows the allocation of tasks to different processors on one Odyssey board. Processor 0 is the Preprocessor, Processors 1 and 2 are the Word Hypothesizers and Processor 3 is the Sentence Hypothesizer. Note that all word hypothesizers operate on the same data from the preprocessor, and communicate with a single sentence hypothesizer. A single Odyssey is capable of recognizing about 100 words. Each additional board is capable of addressing 200 words each.

In designing real-time systems one tends to optimize the entire software. Optimization of real-time software, though desirable, may not necessarily be practical. The resulting increase in processing efficiency does not justify the effort required to optimize *all* the code. It is often found that there are only a few sections of code where a large percentage of the total processing time is spent. Hence, efforts should he directed towards optimizing only these small sections of the code. For example, in the Word Hypothesizer module, it was found that 90 % of the time was spent in the distance measuring routine that compared the input speech with the stored references. Consequently only this module was optimized.

Multi-processor software should be designed so that it can be easily debugged. As with most computer systems the design and specification of a multiprocessor system is done top-down and debugged bottom-up. Thus it is important that one he able to debug the module associated with each processor individually. In the GDCWR implementation, each processor module is designed to communicate via I/O buffers. During the debug process, the input huffer is filled with canned data and the processor is made to execute its function. The output buffer can then be examined for correctness. Using this technique each processor module can be tested prior to integration of the entire application.

Performance Testing

The performance of a connected word recognizer is extremely difficult to quantify because of the lack of accepted data hase and measurement standards. However, Texas Instruments has done a limited amount of testing on this algorithm using an internally developed connected digit data-base. The data used to test the algorithm consisted of 20 speakers reading 5-digit strings. A total of 2000 strings were tested. Two application scenarios are of interest – those applications where the length of the digit sequence is unknown and those (like telephone numbers for example) where the length of the sequence is known. The results of the test are summarized below :

UNKNOWN LENGTH :	5.2% sentence error rate
	1.1% word error rate
KNOWN LENGTH :	3.4% sentence error rate
	0.7% word error rate.

Note that the word error rate for digit strings of known length approaches that achieved for the hest isolated word systems.

Conclusions

We have presented the issues involved in partitioning and allocating tasks in a multiprocessor environment and have discussed in detail the implementation of a connected word recognizer on the Odyssey/Explorer system. Each word hypothesizer is capable of addressing about 50 words providing a 100 word capahility for the first Odyssey board and 200 words for each additional board.

References

[1] P.K.Rajasekaran and G.R.Doddington *Robust Speech Recognition:Initial Results and Progress*, Proceedings of the Darpa Speech Recognition Workshop, Feb. 1986, pp. 73 -80.

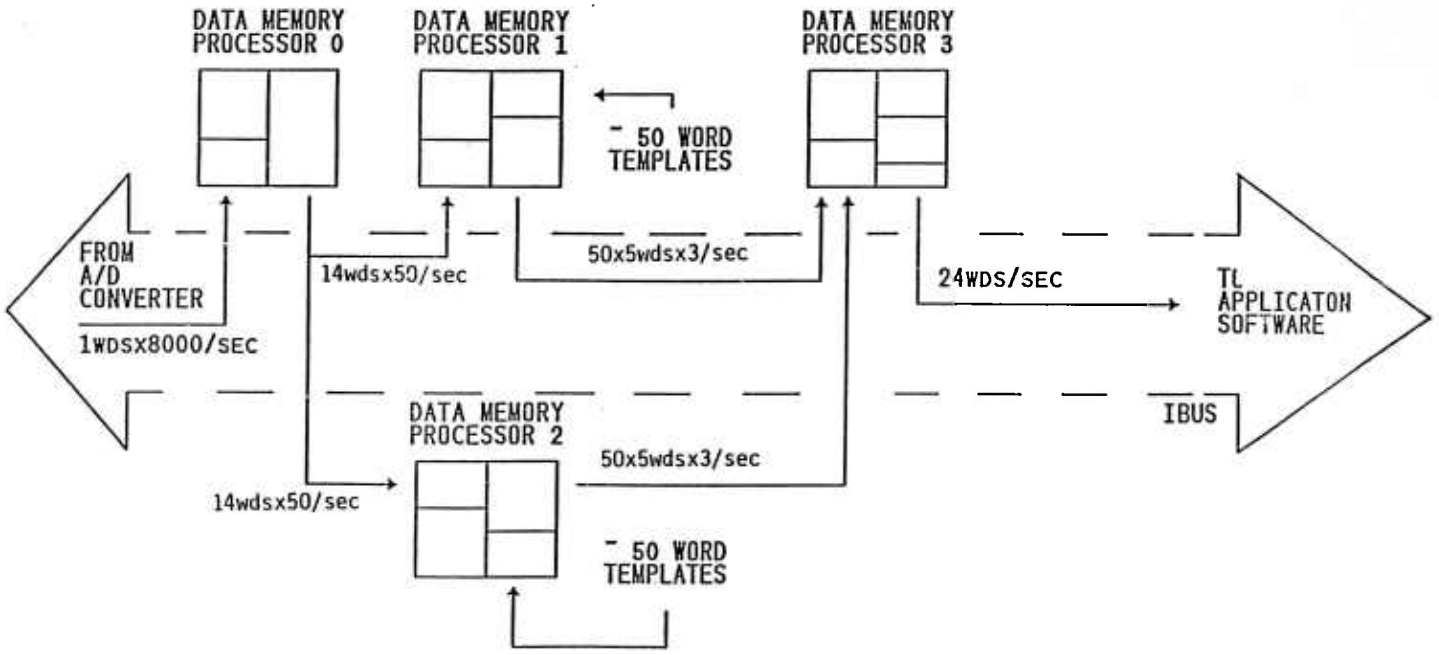
[2] W. Gass, R. Tarrant, and G. Doddington, *A Parallel Signal Processor System*, Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, April 1986, pp. 2887 - 2890.

[3] *TMS32020 User's Guide*, Texas Instruments, 1985.

[4] *Explorer Technical Summary*, Texas Instruments, 1986.

[5] M.L. McMahan and R.B. Price, *Grammar Driven Connected Word Recognition on the TI-SPEECH board*, Proceedings of Speech Tech , April 1986, pp. 88 - 91.

[6] W.W.Chu, L.J.Holloway et.al., *Task Allocation in Distributed Data Processing*, Computer,vol.13,pp.57-62 and 64-69,Nov.1980.



- Multiprocessor speech recognizer implementation
- Grammar-driven 100 word vocabulary using continuous speech
- Bus loading approx. 10924 transfers/second

Figure 3: Allocation of Partitioned Modules.